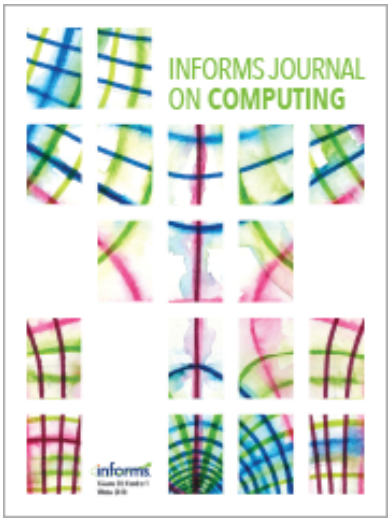


INFORMS Journal on Computing

Publication details, including instructions for authors and subscription information:
http://pubsonline.informs.org

Structured Robust Submodular Maximization: Offline and Online Algorithms

Alfredo Torrico, Mohit Singh, Sebastian Pokutta, Nika Haghtalab, Joseph (Seffi) Naor, Nima Anari

To cite this article:
Alfredo Torrico, Mohit Singh, Sebastian Pokutta, Nika Haghtalab, Joseph (Seffi) Naor, Nima Anari (2021) Structured Robust Submodular Maximization: Offline and Online Algorithms. INFORMS Journal on Computing 33(4):1590-1607. https://doi.org/10.1287/ijoc.2020.0998

http://pubsonline.informs.org/journal/ijoc

INFORMS JOURNAL ON COMPUTING
Vol. 33, No. 4, Fall 2021, pp. 1590–1607
ISSN 1091-9856 (print), ISSN 1526-5528 (online)

Structured Robust Submodular Maximization: Offline and Online Algorithms

Alfredo Torrico,[a] Mohit Singh,[b] Sebastian Pokutta,[c,d] Nika Haghtalab,[e] Joseph (Seffi) Naor,[f] Nima Anari[g]

[a] Polytechnique Montréal, University of Montreal, Montreal, Quebec H3T 1J4, Canada; [b] H. Milton Stewart School of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta, Georgia 30332; [c] Berlin Institute of Technology, 10623 Berlin, Germany; [d] Zuse Institute Berlin, 14195 Berlin, Germany; [e] Department of Computer Science, Cornell University, Ithaca, New York 14853; [f] Technion—Israel Institute of Technology, Haifa 3200003, Israel; [g] Computer Science Department, Stanford University, Stanford, California 94305

Contact: alfredo.torrico-palacios@polymtl.ca, https://orcid.org/0000-0002-9695-9018 (AT); mohit.singh@isye.gatech.edu (MS); pokutta@zib.de (SP); nika@cs.cornell.edu (NH); naor@cs.technion.ac.il (J(S)N); anari@cs.stanford.edu (NA)

Received: August 1, 2019
Revised: March 8, 2020; June 6, 2020
Accepted: June 17, 2020
Published Online in Articles in Advance: February 18, 2021

https://doi.org/10.1287/ijoc.2020.0998

Abstract. Constrained submodular function maximization has been used in subset selection problems such as selection of most informative sensor locations. Although these models have been quite popular, the solutions obtained via this approach are unstable to perturbations in data defining the submodular functions. Robust submodular maximization has been proposed as a richer model that aims to overcome this discrepancy as well as increase the modeling scope of submodular optimization. In this work, we consider robust submodular maximization with structured combinatorial constraints and give efficient algorithms with provable guarantees. Our approach is applicable to constraints defined by single or multiple matroids and knapsack as well as distributionally robust criteria. We consider both the offline setting where the data defining the problem are known in advance and the online setting where the input data are revealed over time. For the offline setting, we give a general (nearly) optimal bicriteria approximation algorithm that relies on new extensions of classical algorithms for submodular maximization. For the online version of the problem, we give an algorithm that returns a bicriteria solution with sublinear regret.
Summary of Contribution: Constrained submodular maximization is one of the core areas in combinatorial optimization with a wide variety of applications in operations research and computer science. Over the last decades, both communities have been interested on the design and analysis of new algorithms with provable guarantees. Sensor location, influence maximization and data summarization are some of the applications of submodular optimization that lie at the intersection of the aforementioned communities. Particularly, our work focuses on optimizing several submodular functions simultaneously. We provide new insights and algorithms to the offline and online variants of the problem which significantly expand the related literature. At the same time, we provide a computational study that supports our theoretical results.

History: Accepted by Andrea Lodi, Area Editor for Design and Analysis of Algorithms—Discrete.
Funding: This work was supported in part by the U.S. National Science Foundation (NSF) [Grant CCF-1717947], an NSF Career Award [Grant CMMI-1452463], an IBM PhD fellowship, and a Microsoft Research PhD fellowship.
Supplemental Material: The online supplement is available at https://doi.org/10.1287/ijoc.2020.0998.

Keywords: submodular optimization • robust optimization • matroid constraints • greedy algorithm • online learning

1. Introduction

Constrained submodular function maximization has seen significant progress in recent years in the design and analysis of new algorithms with guarantees (Sviridenko 2004, Calinescu et al. 2011, Buchbinder and Feldman 2016, Ene and Nguyen 2016), as well as numerous applications, especially in constrained subset selection problems (Krause and Guestrin 2005; Krause et al. 2008a,b; 2009; Lin and Bilmes 2009; Powers et al. 2016a), and more broadly machine learning. A typical example is the problem of picking a subset of candidate sensor locations for spatial monitoring of certain phenomena such as temperature, pH values, humidity, and so on (Krause et al. 2008b). Here the goal is typically to find sensor locations that achieve the most coverage or give the most information about the observed phenomena. Submodularity naturally captures the decreasing marginal gain in the coverage or the information acquired about relevant phenomena by using more sensors (Das and Kempe 2008). Although submodular optimization offers an attractive model for such scenarios, there are a few key shortcomings, which motivated robust submodular optimization in the cardinality case (Krause et al. 2008b) so as to optimize against several functions simultaneously.

1. The sensors are typically used to measure various parameters at the same time. Observations for these parameters need to be modeled via different submodular functions.

2. Many of the phenomena being observed are nonstationary and highly variable in certain locations. To obtain a good solution, a common approach is to use different submodular functions to model different spatial regions.

3. The submodular functions are typically defined using data obtained from observations, and imprecise information can lead to unstable optimization problems. Thus, there is a desire to compute solutions that are robust to perturbations of the submodular functions.

Given the computational complexity of optimizing several functions simultaneously, Krause et al. (2008b) motivated a bicriteria approach. In simple words, to obtain provable guarantees, one needs to trade off the quality of the solution measured by its objective value with the *size* of the solution. In the case of a single cardinality constraint, Krause et al. (2008b) propose a natural relaxation that consists of allowing more elements in the final set, that is, violating the feasibility constraint. However, to obtain provable guarantees for more general combinatorial constraints, relaxing the size of feasible sets is not enough.

Our main contribution is the development of new algorithms with provable guarantees for robust submodular optimization under a large class of combinatorial constraints. These include partition constraints, where local cardinality constraints are placed on disjoint parts of the ground set. More generally, we consider matroid and knapsack constraints.

We provide bicriteria approximations that trade off the approximation factor with the size of the solution, measured by the number ℓ of feasible sets $\{S_i\}_{i\in[\ell]}$ whose union constitutes the final solution S. Although this might be nonintuitive at first, it turns out that the union of feasible sets corresponds to an appropriate analog of the single cardinality constraint. Some special cases of interest include

1. *Partition constraints.* Given a partition of the candidate sensor locations, the feasible sets correspond to subsets that satisfy a cardinality constraint on each part of the partition. The union of feasible sets here corresponds to relaxing the cardinality constraints separately for each part. This results in a stronger guarantee than relaxing the constraint globally, as would be the case in the single cardinality constraint case.

2. *Gammoid.* Given a directed graph and a subset of nodes T, the feasible sets correspond to subsets S that can reach T via disjoint paths in the graph. Gammoids appear in flow-based models, for example, in reliable routing. The union of feasible sets now corresponds to sets S that can reach T via paths such that each vertex appears in few paths.

We consider both offline and online versions of the problem, where the data are either known a priori or are revealed over time, respectively. For the offline version of the problem, we provide a general procedure that iteratively uses any standard algorithm for submodular maximization to produce a solution that is a union of multiple feasible sets. The analysis relies on known insights about the performance of the classic greedy algorithm when it is used for the cardinality constraint. For the online case, we introduce new technical ingredients that might be broadly applicable in online robust optimization. Our work significantly expands on previous work on robust submodular optimization that focused on a single cardinality constraint (Krause et al. 2008b). Moreover, our work substantially differs from its proceeding version (Anari et al. 2019) because (1) we provide a general and flexible framework to design an offline bicriteria algorithm, and (2) we give significant implementation improvements and support our theoretical results via an exhaustive computational study on two real-world applications.

1.1. Problem Formulation

As we describe next, we study offline and online variations of robust submodular maximization under structured combinatorial constraints. Although our results hold for more general constraints, we focus our attention first on matroid constraints that generalize the partition as well as the gammoid structural constraints mentioned earlier. We discuss extensions to other class of constraints in Section 4.

Consider a nonnegative set function $f : 2^V \to \mathbb{R}_+$. We denote the marginal value for any subset $A \subseteq V$ and $e \in V$ by $f_A(e) := f(A + e) - f(A)$, where $A + e := A \cup \{e\}$. Function f is submodular if and only if it satisfies the diminishing-returns property. Namely, for any $e \in V$ and $A \subseteq B \subseteq V\backslash\{e\}$, $f_A(e) \geq f_B(e)$. We say that f is monotone if for any $A \subseteq B \subseteq V$ we have $f(A) \leq f(B)$. Most of our results are concerned with optimization of monotone submodular functions.

A natural class of constraints considered in submodular optimization is matroid constraints. For a ground set V and a family of sets $\mathcal{I} \subseteq 2^V$, $\mathcal{M} = (V, \mathcal{I})$ is a matroid if (1) for all $A \subset B \subseteq V$, if $B \in \mathcal{I}$, then $A \in \mathcal{I}$, and (2) for all $A, B \in \mathcal{I}$ with $|A| < |B|$, there is $e \in B \setminus A$ such that $A \cup \{e\} \in \mathcal{I}$. Sets in such a family $\mathcal{I}$ are called *independent sets* or, simply put, *feasible sets* for the purpose of optimization. Maximal independent sets are called *bases*. Finally, the rank of a matroid is the maximum size of an independent set in the matroid.

The classical problem of maximizing a single monotone submodular function $f : 2^V \to \mathbb{R}_+$ under a matroid constraint $\mathcal{M} = (V, \mathcal{I})$ is formally stated as $\max_{A\in\mathcal{I}} f(A)$.

Throughout this paper, we say that a polynomial-time algorithm $\mathcal{A}$ achieves a $(1-\beta)$ approximation factor for this problem if it returns a feasible solution $S \in \mathcal{I}$ such that $f(S) \geq (1-\beta)\cdot \max_{A\in\mathcal{I}} f(A)$. When $\mathcal{A}$ is randomized, we say that $\mathcal{A}$ achieves a $(1-\beta)$-approximation in expectation if $\mathbb{E}[f(S)] \geq (1-\beta)\cdot \max_{A\in\mathcal{I}} f(A)$. The well-known standard greedy algorithm (Nemhauser et al. 1978) is an example of $\mathcal{A}$ with $\beta = 1/2$ (Fisher et al. 1978). We will denote by time($\mathcal{A}$) the running time of $\mathcal{A}$.

In this work, we consider the robust variation of the previous submodular maximization problem. That is, for a matroid $\mathcal{M} = (V, \mathcal{I})$, and a given collection of k monotone submodular functions $f_i : 2^V \to \mathbb{R}_+$ for $i \in [k]$, our goal is to efficiently select a set S that maximizes $\min_{i\in[k]} f_i(S)$. We define a $(1-\epsilon)$-approximately optimal solution S as follows:

$$\min_{i\in[k]} f_i(S) \geq (1-\epsilon)\cdot \max_{A\in\mathcal{I}} \min_{i\in[k]} f_i(A). \tag{1}$$

We also consider the online variation of the previous optimization problem in the presence of an adversary. In this setting, we are given a fixed matroid $\mathcal{M} = (V, \mathcal{I})$. At each time step $t \in [T]$, we choose a set S^t. An adversary then selects a collection of k monotone submodular functions $\{f_i^t\}_{i\in[k]} : 2^V \to [0,1]$. We receive a reward of $\min_{i\in[k]} \mathbb{E}[f_i^t(S^t)]$, where the expectation is taken over any randomness in choosing S^t. We can then use the knowledge of the adversary's actions, that is, oracle access to $\{f_i^t\}_{i\in[k]}$, in our future decisions. We consider nonadaptive adversaries whose choices $\{f_i^t\}_{i\in[k]}$ are independent of S^τ for $\tau < t$. In other words, an adversarial sequence of functions $\{f_i^1\}_{i\in[k]}, \ldots, \{f_i^T\}_{i\in[k]}$ is chosen upfront without being revealed to the optimization algorithm. Our goal is to design an algorithm that maximizes the total payoff $\sum_{t\in[T]} \min_{i\in[k]} \mathbb{E}[f_i^t(S^t)]$. Thus, we would like to obtain a cumulative reward that competes with that of the fixed set $S \in \mathcal{I}$ we should have played had we known all the functions f_i^t in advance, that is, compete with $\max_{S\in\mathcal{I}} \sum_{t\in[T]} \min_{i\in[k]} f_i^t(S)$. As in the offline optimization problem, we also consider competing with $(1-\epsilon)$ fraction of the previous benchmark. In this case, $\mathbf{Regret}_{1-\epsilon}(T)$ denotes how far we are from this goal. That is,

$$\mathbf{Regret}_{1-\epsilon}(T) = (1-\epsilon)\cdot \max_{S\in\mathcal{I}} \sum_{t\in[T]} \min_{i\in[k]} f_i^t(S) - \sum_{t\in[T]} \min_{i\in[k]} \mathbb{E}\left[f_i^t(S^t)\right]. \tag{2}$$

We desire algorithms whose $(1-\epsilon)$-regret is sublinear in T. That is, we get arbitrarily close to a $(1-\epsilon)$ fraction of the benchmark as $T \to \infty$.

The offline (Equation (1)) or online (Equation (2)) variations of robust monotone submodular functions are known to be nondeterministic polynomial-time hard (NP-hard) to approximate to any polynomial factor when the algorithm's choices are restricted to the family of independent sets $\mathcal{I}$ (Krause et al. 2008b). Therefore, to obtain any reasonable approximation guarantee, we need to relax the algorithm's constraint set. Such an approximation approach is called a *bicriteria approximation scheme*, in which the algorithm outputs a set with a nearly optimal objective value while ensuring that the set used is the union of only a few independent sets in $\mathcal{I}$. More formally, to get a $(1-\epsilon)$-approximate solutions, we may use a set S where $S = S_1 \cup \cdots \cup S_\ell$ such that $S_1, \ldots, S_\ell \in \mathcal{I}$ and ℓ is a function of $\frac{1}{\epsilon}$ and other parameters. Because the output set S is possibly infeasible, we define the violation ratio ν as the minimum number of feasible sets whose union is S. To exemplify this, consider partition constraints: Here we are given a partition $\{P_1, \ldots, P_q\}$ of the ground set, and the goal is to pick a subset that includes at most b_j elements from part P_j for each j. Then the union of ℓ feasible sets has at most $\ell \cdot b_j$ elements in each part. In our example, the violation ratio corresponds to $\nu = \max_{j\in[q]} \lceil |S \cap P_j| / b_j \rceil$.

1.2. Our Results and Contributions

We present (nearly tight) bicriteria approximation algorithms for the offline and online variations of robust monotone submodular optimization under matroid constraints. Throughout this paper, we assume that the matroid is accessible via an independence oracle and that the submodular functions are accessible via a value oracle. Moreover, we use $\log_a$ to denote a logarithm with base a (when the subscript is not explicit, we assume that it is base 2; that is, $\log := \log_2$) and ln to denote the natural logarithm.

For the offline setting of the problem, we obtain the following general result.

Theorem 1. *Consider a polynomial-time $(1-\beta)$-approximation algorithm $\mathcal{A}$ for the problem of maximizing a single monotone submodular function subject to a matroid constraint. Then, for the offline robust submodular optimization problem (1), for any $0 < \epsilon < 1$, there is a polynomial-time bicriteria algorithm that uses ext-$\mathcal{A}$ as a subroutine, runs in*

$$O\left(\text{time}(\mathcal{A})\cdot \log_{1/\beta}\left(\frac{k}{\epsilon}\right)\cdot \log(n) \cdot \min\left\{\frac{nk}{\epsilon}, \log_{1+\epsilon}\left(\max_{e,j} f_j(e)\right)\right\}\right)$$

time, and returns a set S^{ALG} such that

$$\mathbb{E}\left[\min_{i\in[k]} f_i\left(S^{\text{ALG}}\right)\right] \geq (1-\epsilon)\cdot \max_{S\in\mathcal{I}} \min_{j\in[k]} f_j(S),$$

where the expectation is taken over any randomization of $\mathcal{A}$, $S^{\text{ALG}} = S_1 \cup \cdots \cup S_\ell$, with $\ell = \lceil \log_{1/\beta} \frac{k}{\epsilon} \rceil$ and $S_1, \ldots, S_\ell \in \mathcal{I}$.

The subroutine ext-$\mathcal{A}$ that achieves this result is an extended version of the algorithm $\mathcal{A}$. Because $\mathcal{A}$ outputs a feasible set, ext-$\mathcal{A}$ reuses $\mathcal{A}$ in an iterative scheme so that it generates a small family of feasible sets whose union achieves the $(1-\epsilon)$-guarantee. The argument is reminiscent of a well-known fact for submodular function maximization under cardinality constraints: Letting the standard greedy algorithm run longer results in better approximations at the expense of violating the cardinality constraint. Our extended algorithm ext-$\mathcal{A}$ works in a similar spirit, but it iteratively produces feasible sets in the matroid. This framework generalizes the idea presented in Anari et al. (2019). We emphasize that our procedure does not correspond to an extension of the algorithm for the cardinality constraint presented in Krause et al. (2008b). The natural extension of their algorithm to a single matroid constraint would be to run an algorithm $\mathcal{A}$ over a larger feasibility constraint. However, this approach does not provide any bicriteria approximation. The main challenge for matroid constraints is to define an appropriate notion of violation. In our work, we measure violation by the number of independent sets needed to cover the given set, in contrast to Krause et al. (2008b), who define it in terms of cardinality. Our subroutine ext-$\mathcal{A}$ constructs a family of independent sets that is pivotal to obtain provable guarantees.

A natural candidate for $\mathcal{A}$ is the standard greedy algorithm, hereafter referred to as *Greedy*. Because Greedy achieves a 1/2 approximation factor (Fisher et al. 1978), the number of feasible sets needed is $\ell = \lceil \log_2 \frac{k}{\epsilon} \rceil$ (Anari et al. 2019). Unfortunately, Greedy's running time $O(n \cdot r)$ depends on the rank of the matroid r. This can be computationally inefficient when r is sufficiently large. To improve this, in Section 2.1, we propose an extended version of the threshold greedy algorithm introduced by Badanidiyuru and Vondrák (2014). The main advantage of the threshold greedy algorithm, further referred as *ThGreedy*, is its running time, which does not depend on the rank of the matroid. Formally, we obtain the following corollary.

Corollary 1. *For the offline robust submodular optimization problem (1), for any $0 < \epsilon, \delta < 1$, there is a polynomial-time bicriteria algorithm that uses ext-ThGreedy as a subroutine, runs in*

$$O\left(\frac{n}{\delta} \cdot \log\left(\frac{n}{\delta}\right) \cdot \log_2\left(\frac{k}{\epsilon}\right) \cdot \log(n) \cdot \min\left\{\frac{nk}{\epsilon}, \log_{1+\epsilon}\left(\max_{e,j} f_j(e)\right)\right\}\right)$$

time, and returns a set S^{ALG} such that

$$\min_{i\in[k]} f_i\left(S^{\mathrm{ALG}}\right) \geq (1-\epsilon) \cdot \max_{S\in\mathcal{I}} \min_{j\in[k]} f_j(S),$$

where $S^{\mathrm{ALG}} = S_1 \cup \cdots \cup S_\ell$ with $\ell = \lceil \log_2 \frac{k}{\epsilon} \rceil$ and $S_1, \ldots, S_\ell \in \mathcal{I}$.

To achieve a tight bound on the size of the family ℓ, we use an improved version of the continuous greedy algorithm (Vondrák 2008, Badanidiyuru and Vondrák 2014) as the inner algorithm $\mathcal{A}$. Because the continuous greedy algorithm achieves a $1 - 1/e$ approximation ratio, we need $\ell = \lceil \ln \frac{k}{\epsilon} \rceil$ feasible sets to achieve a $1-\epsilon$ fraction of the true optimum, which matches the hardness result presented in Krause et al. (2008b). This bicriteria algorithm is much simpler than the one presented in Anari et al. (2019). We present the main offline results and the corresponding proofs in Section 2.

One might hope that similar results can be obtained even when functions are nonmonotone (but still submodular). As we show in Section 2.2, this is not possible. To support our theoretical guarantees, we provide an exhaustive computational study in Section 2.3. In this section, we observe that the main computational bottleneck of the bicriteria algorithms is to certify the near-optimality of the output solution. To solve this, we present significant implementation improvements such as lazy evaluations and an early stopping criterion, which empirically show how the computational cost can be drastically improved.

Our offline approach is quite flexible in the sense that Theorem 1 uses an arbitrary algorithm $\mathcal{A}$. This allows us to consider different algorithms in the literature of submodular optimization and further extend our results to other classes of constraints, such as knapsack constraints and multiple matroids. We describe these extensions in Section 4.

A natural question is whether our algorithm can be carried over into the online setting, where functions are revealed over time. For the online variant, we present the following result.

Theorem 2. *For the online robust submodular optimization problem (2), for any $0 < \epsilon < 1$, there is a randomized polynomial-time algorithm that returns a set S^t for each stage $t \in [T]$, and thus we get the following:*

$$\sum_{t\in[T]} \min_{i\in[k]} \mathbb{E}\left[f_i^t\left(S^t\right)\right] \geq (1-\epsilon) \cdot \max_{S\in\mathcal{I}} \sum_{t\in[T]} \min_{i\in[k]} f_i^t(S) - O\left((1-\epsilon) n^{\frac{5}{4}} \sqrt{T}\right),$$

where $S^t = S_1^t \cup \cdots \cup S_\ell^t$, with $\ell = \lceil \ln \frac{1}{\epsilon} \rceil$ and $S_1^t, \ldots, S_\ell^t \in \mathcal{I}$.

We remark that the guarantee of Theorem 2 holds with respect to the minimum of $\mathbb{E}[f_i^t(S^t)]$, as opposed to the guarantee of Theorem 1, which directly bounds the minimum of $f_i(S)$. Therefore, the solution for the online algorithm is a union of only $\lceil \ln \frac{1}{\epsilon} \rceil$ independent sets, in contrast to the offline solution, which is the union of $\lceil \log \frac{k}{\epsilon} \rceil$ independent sets. The main challenge

in the online algorithm is to deal with nonconvexity and nonsmoothness resulting from submodularity exacerbated by the robustness criteria. Our approach to coping with the robustness criteria is to use the soft-min function $-\frac{1}{\alpha}\ln\sum_{i\in[k]} e^{-\alpha g_i}$, defined for a collection of smooth functions $\{g_i\}_{i\in[k]}$ and a suitable parameter $\alpha > 0$. This function is also known as *log-sum-exp*; for some of its properties and applications, we refer the interested reader to Calafiore and El Ghaoui (2014). Although the choice of the specific soft-min function is seemingly arbitrary, one feature is crucial for us: Its gradient is a convex combination of the gradients of the g_i's. Using this observation, we use parallel instances of the follow-the-perturbed-leader (FPL) algorithm presented by Kalai and Vempala (2005), one for each discretization step in the continuous greedy algorithm. We believe that the algorithm might be of independent interest to perform online learning over a minimum of several functions, a common feature in robust optimization. The main result and its proof appear in Section 3.

1.3. Related Work

Building on the classical work of Nemhauser et al. (1978), constrained submodular maximization problems have seen much progress recently (see, e.g., Chekuri et al. 2010, Calinescu et al. 2011, Buchbinder et al. 2014, 2016). Robust submodular maximization generalizes submodular function maximization under a matroid constraint for which a $(1-\frac{1}{e})$-approximation is known (Calinescu et al. 2011) and is optimal. The problem has been studied for constant k by Chekuri et al. (2010), who give a $(1-\frac{1}{e}-\epsilon)$-approximation algorithm with running time $O(n^{\frac{k}{\epsilon}})$. Closely related to our problem is the submodular cover problem, where we are given a submodular function f and a target $b \in \mathbb{R}_+$, and the goal is to find a set S of minimum cardinality such that $f(S) \geq b$. A simple reduction shows that robust submodular maximization under a cardinality constraint reduces to the submodular cover problem (Krause et al. 2008b). Wolsey (1982) shows that the greedy algorithm gives an $O(\ln\frac{n}{\epsilon})$-approximation, where the output set S satisfies $f(S) \geq (1-\epsilon)b$. Krause et al. (2008b) use this approximation to build a bicriteria algorithm for the cardinality case that achieves tight bounds. This approach falls short of achieving a bicriteria approximation when the problem is defined over a matroid. A natural extension of this approach to matroid constraints would be to run a single algorithm $\mathcal{A}$ over a larger feasibility constraint. However, this procedure does not provide any guarantee. Therefore, the main challenge is to define an appropriate notion of violation. In this work, we measure violation not by cardinality, as in Krause et al. (2008b), but by the number of independent sets needed to cover the given set. Our subroutine ext-$\mathcal{A}$ picks one feasible set at a time, and this is crucial to obtain provable guarantees. Powers et al. (2016b) consider the same robust problem with matroid constraints. However, they take a different approach by presenting a bicriteria algorithm that outputs a feasible set that is good only for a fraction of the k monotone submodular functions. A deletion-robust submodular optimization model is presented in Krause et al. (2008b), which is later studied by Orlin et al. (2016), Bogunovic et al. (2017), and Kazemi et al. (2018). Influence maximization (Kempe et al. 2003) in a network has been a successful application of submodular maximization, and recently, He and Kempe (2016) and Chen et al. (2016) study the robust influence maximization problem. Robust optimization for nonconvex objectives (including submodular functions) has been also considered by Chen et al. (2017), but with weaker guarantees than ours because of the extended generality. Specifically, their algorithm outputs $\frac{r\log k}{\epsilon^2\mathrm{OPT}}$ feasible sets whose union achieves a factor of $(1-1/e-\epsilon)$. Finally, Wilder (2017) studies a similar problem in which the set of feasible solutions is the set of all distributions over independent sets of a matroid. In particular, for our setting, Wilder (2017) gives an algorithm that outputs $O(\frac{\log k}{\epsilon^3})$ feasible sets whose union obtains $(1-1/e)^2$ fraction of the optimal solution. Our results are stronger than the ones obtained by Chen et al. (2017) and Wilder (2017) because we provide the same guarantees using the union of fewer feasible sets. Other variants of the robust submodular maximization problem are studied by Mitrovic et al. (2018) and Staib et al. (2018).

There has been some prior work on online submodular function maximization that we briefly review here. Streeter and Golovin (2008) study the budgeted maximum submodular coverage problem and consider several feedback cases (denote B a integral bound for the budget): In the full-information case, a $(1-\frac{1}{e})$-expected regret of $O(\sqrt{BT\ln n})$ is achieved, but the algorithm uses B experts, which may be very large. In a follow-up work, Golovin et al. (2014) study the online submodular maximization problem under partition constraints, and then they generalize it to general matroid constraints. For the latter algorithm, the authors present an online version of the continuous greedy algorithm, which relies on the FPL algorithm of Kalai and Vempala (2005), and obtain a $(1-\frac{1}{e})$-expected regret of $O(\sqrt{T})$. Similar to this approach, our bicriteria online algorithm will also use the FPL algorithm as a subroutine. Recent results on other variants of the online submodular maximization problem are studied in Soma (2019) and Zhang et al. (2019).

2. The Offline Case

In this section, we consider offline robust optimization (Equation (1)) under matroid constraints.

2.1. General Offline Algorithm and Analysis

In this section, we present a general procedure to achieve a (nearly) tight bicriteria approximation for the problem of interest and prove Theorem 1.

First, consider a nonnegative monotone submodular function $g : 2^V \to \mathbb{R}_+$, a matroid $\mathcal{M} = (V, \mathcal{I})$, and a polynomial-time $(1-\beta)$-approximation algorithm $\mathcal{A}$ for the problem of maximizing g over $\mathcal{M}$. Formally, in the deterministic case, $\mathcal{A}$ outputs a feasible set $S \in \mathcal{I}$ such that $g(S) \geq (1-\beta) \cdot \max_{A \in \mathcal{I}} g(A)$. If $g(\emptyset) \neq 0$, then we define a new function $g' : 2^V \to \mathbb{R}_+$ as $g'(A) := g(A) - g(\emptyset)$, which remains being monotone and submodular. The approximation guarantee in this case is $g(S) - g(\emptyset) \geq (1-\beta) \cdot \max_{A \in \mathcal{I}} \{g(A) - g(\emptyset)\}$. When $\mathcal{A}$ is a randomized algorithm, we say that $\mathcal{A}$ achieves a $(1-\beta)$ factor in expectation if $\mathcal{A}$ outputs a random feasible set $S \in \mathcal{I}$ such that $\mathbb{E}[g(S) - g(\emptyset)] \geq (1-\beta) \cdot \max_{A \in \mathcal{I}} \{g(A) - g(\emptyset)\}$. We define in Algorithm 1 our main procedure ext-$\mathcal{A}$ as an extended version of $\mathcal{A}$ that runs iteratively $\ell \geq 1$ times.

Algorithm 1 (General Extended Algorithm for Submodular Optimization ext-$\mathcal{A}$)

Input: $\ell \geq 1$, a monotone submodular function $g : 2^V \to \mathbb{R}_+$, a matroid $\mathcal{M} = (V, \mathcal{I})$, and algorithm $\mathcal{A}$.

Output: sets $S_1, \ldots, S_\ell \in \mathcal{I}$.

1: **for** $\tau = 1, \ldots, \ell$, **do**
2: Define $\tilde{g}(S) = g(S \cup \cup_{j=1}^{\tau-1} S_j)$.
3: $S_\tau \leftarrow \mathcal{A}(\tilde{g}, \mathcal{M})$.

Note that function $\tilde{g}$, defined in line 2 of Algorithm 1, is also monotone and submodular. Observe that we can recover algorithm $\mathcal{A}$ simply by considering $\ell = 1$ in ext-$\mathcal{A}$. More important, we obtain the following guarantee for ext-$\mathcal{A}$.

Theorem 3. *Consider a monotone submodular function $g : 2^V \to \mathbb{R}_+$ with $g(\emptyset) = 0$, a matroid $\mathcal{M} = (V, \mathcal{I})$, and a polynomial-time $(1-\beta)$-approximation algorithm $\mathcal{A}$ for the problem of maximizing g over $\mathcal{M}$. For any $\ell \geq 1$, Algorithm 1 returns sets $S_1, \ldots, S_\ell$ such that*

$$\mathbb{E}\left[g\left(\bigcup_{\tau=1}^{\ell} S_\tau \right) \right] \geq \left(1 - \beta^\ell\right) \cdot \max_{S \in \mathcal{I}} g(S),$$

where the expectation is taken over any randomization of $\mathcal{A}$ when choosing $S_1, \ldots, S_\ell$.

Proof. Let us assume that $\mathcal{A}$ is deterministic. In the randomized case, the proof follows similarly by taking the corresponding expectations. From the first iteration of Algorithm 1 and using the guarantees of $\mathcal{A}$, we conclude that $g(S_1) - g(\emptyset) \geq (1-\beta) \cdot \max_{S \in \mathcal{I}} \{g(S) - g(\emptyset)\}$. We use the previous statement to prove our theorem by induction. For $\tau = 1$, the claim follows directly. Consider any $\ell \geq 2$. Observe that the algorithm in iteration $\tau = \ell$ is exactly algorithm $\mathcal{A}$ run on submodular function $\tilde{g} : 2^V \to \mathbb{R}_+$, where $\tilde{g}(S) := g(S \cup \cup_{j=1}^{\ell-1} S_j)$. This procedure returns S_ℓ such that $\tilde{g}(S_\ell) - \tilde{g}(\emptyset) \geq (1-\beta) \cdot \max_{S \in \mathcal{I}} \{\tilde{g}(S) - \tilde{g}(\emptyset)\}$, which implies that

$$g\left(\bigcup_{\tau=1}^{\ell} S_\tau \right) - g\left(\bigcup_{\tau=1}^{\ell-1} S_\tau \right) \geq \left(1 - \beta\right) \cdot \max_{S \in \mathcal{I}} \left\{ g(S) - g\left(\bigcup_{\tau=1}^{\ell-1} S_\tau \right) \right\}.$$

By induction, we know that $g(\cup_{\tau=1}^{\ell-1} S_\tau) \geq (1 - \beta^{\ell-1}) \cdot \max_{S \in \mathcal{I}} g(S)$. Thus, we obtain the following:

$$\begin{aligned} g\left(\bigcup_{\tau=1}^{\ell} S_\tau \right) &\geq \left(1 - \beta\right) \cdot \max_{S \in \mathcal{I}} g(S) + \beta \cdot g\left(\bigcup_{\tau=1}^{\ell-1} S_\tau \right) \\ &\geq \left(1 - \beta^\ell\right) \cdot \max_{S \in \mathcal{I}} g(S). \quad \square \end{aligned}$$

We now apply Theorem 3 for the robust submodular problem, in which we are given monotone submodular functions $f_i : 2^V \to \mathbb{R}_+$ with $f_i(\emptyset) = 0$ for $i \in [k]$. Our main bicriteria algorithm consists of two consecutive steps: (1) Get an estimate γ of the value of the optimal solution OPT, and (2) apply subroutine ext-$\mathcal{A}$ to a convenient function depending on γ. Formally, we obtain an estimate γ on the value of the optimal solution $\mathrm{OPT} := \max_{S \in \mathcal{I}} \min_{i \in [k]} f_i(S)$ via a binary search. For the purpose of the proof of Theorem 1, given parameter $\epsilon > 0$, let us assume that γ has relative error $1 - \frac{\epsilon}{2}$; that is, $(1 - \frac{\epsilon}{2})\mathrm{OPT} \leq \gamma \leq \mathrm{OPT}$. As in Krause et al. (2008b), let $g : 2^V \to \mathbb{R}_+$ be defined for any $S \subseteq V$ as follows:

$$g(S) := \frac{1}{k} \sum_{i \in [k]} \min\{f_i(S), \gamma\}. \tag{3}$$

Observe that $\max_{S \in \mathcal{I}} g(S) = \gamma$ whenever $\gamma \leq \mathrm{OPT}$. Moreover, note that g is also a monotone submodular function. Therefore, the second step of the bicriteria algorithm is to run algorithm ext-$\mathcal{A}$ on the function g to obtain a candidate solution. A more detailed description of the algorithm can be found in Section 2.3.

Proof of Theorem 1. We assume $\mathcal{A}$ to be deterministic. The randomized case can be easily proved by consider the following proof for each realization sequence of $S_1, \ldots, S_\ell$. Consider the family of monotone submodular functions $\{f_i\}_{i \in [k]}$, and define g as in Equation (3) using parameter γ with a relative error of $1 - \frac{\epsilon}{2}$. If we run Algorithm 1 on g with $\ell \geq \lceil \log_{1/\beta} \frac{2k}{\epsilon} \rceil$, we get a

set $S^{\mathrm{ALG}} = S_1 \cup \cdots \cup S_\ell$, where $S_j \in \mathcal{I}$ for all $j \in [\ell]$. Moreover, Theorem 3 implies that

$$g(S^{\mathrm{ALG}}) \geq (1-\beta^\ell) \cdot \max_{S\in\mathcal{I}} g(S) \geq \left(1-\frac{\epsilon}{2k}\right)\cdot\gamma.$$

Now we will prove that $f_i(S^{\mathrm{ALG}}) \geq (1-\frac{\epsilon}{2})\cdot\gamma$ for all $i \in [k]$. Assume by contradiction that there exists an index $i^* \in [k]$ such that $f_{i^*}(S^{\mathrm{ALG}}) < (1-\frac{\epsilon}{2})\cdot\gamma$. Because we know that $\min\{f_i(S^{\mathrm{ALG}}), \gamma\} \leq \gamma$ for all $i \in [k]$, then

$$g(S^{\mathrm{ALG}}) \leq \frac{1}{k}\cdot f_{i^*}(S^{\mathrm{ALG}}) + \frac{k-1}{k}\cdot\gamma < \frac{1-\epsilon/2}{k}\cdot\gamma + \frac{k-1}{k}\cdot\gamma = \left(1-\frac{\epsilon}{2k}\right)\cdot\gamma,$$

contradicting $g(S^{\mathrm{ALG}}) \geq (1-\frac{\epsilon}{2k})\cdot\gamma$. Therefore, we obtain $f_i(S^{\mathrm{ALG}}) \geq (1-\frac{\epsilon}{2})\cdot\gamma \geq (1-\epsilon)\cdot\mathrm{OPT}$ for all $i \in [k]$ as claimed. □

2.1.1. Running-Time Analysis. In this section, we study the running time of the bicriteria algorithm we just presented. To show that a set of polynomial-size values for γ exists such that one of them satisfies $(1-\epsilon/2)\mathrm{OPT} \leq \gamma \leq \mathrm{OPT}$, we simply try $\gamma = nf_i(e)(1-\epsilon/2)^j$ for all $i \in [k]$, $e \in V$, and $j = 0, \ldots, \lceil \ln_{1-\epsilon/2}(1/n)\rceil$. Note that there exists an index $i^* \in [k]$ and a set $S^* \in \mathcal{I}$ such that $\mathrm{OPT} = f_{i^*}(S^*)$. Now let $e^* = \arg\max_{e\in S^*} f_{i^*}(e)$. Because of submodularity and monotonicity, we have $\frac{1}{|S^*|} f_{i^*}(S^*) \leq f_{i^*}(e^*) \leq f_{i^*}(S^*)$. So we can conclude that $1 \geq \mathrm{OPT}/nf_{i^*}(e^*) \geq 1/n$, which implies that $j = \lceil \ln_{1-\epsilon/2}(\mathrm{OPT}/nf_{i^*}(e^*))\rceil$ is in the correct interval, obtaining

$$(1-\epsilon/2)\ \mathrm{OPT} \leq nf_{i^*}(e^*)(1-\epsilon/2)^j \leq \mathrm{OPT}.$$

We remark that the dependency of the running time on ϵ can be made logarithmic by running a binary search on j as opposed to trying all $j = 0, \ldots, \lceil \ln_{1-\epsilon/2}(1/n)\rceil$. This would take at most $\frac{nk}{\epsilon}\cdot\log n$ iterations. We could also say that doing a binary search to get a value up to a relative error of $1-\epsilon/2$ of OPT would take $\log_{1+\epsilon}\mathrm{OPT}$. So we consider the minimum of those two quantities $\min\{\frac{nk}{\epsilon}\cdot\log n, \log_{1+\epsilon}\mathrm{OPT}\}$. Given that Algorithm 1 runs in $\ell\cdot\mathrm{time}(\mathcal{A})$, where $\ell = \lceil\log_{1/\beta}\frac{k}{\epsilon}\rceil$ is the number of rounds, we conclude that the bicriteria algorithm runs in $O(\mathrm{time}(\mathcal{A})\cdot\log_{1/\beta}\frac{k}{\epsilon}\cdot\min\{\frac{nk}{\epsilon}\cdot\log n, \log_{1+\epsilon}\mathrm{OPT}\})$ time.

2.1.2. Two Deterministic Classical Algorithms: Greedy and Local Search. The most natural candidate for $\mathcal{A}$ is Greedy (Nemhauser et al. 1978), which achieves a ratio of 1/2 for the problem of maximizing a single monotone submodular function subject to a matroid constraint (Fisher et al. 1978). In this case, we know that $\mathrm{time(Greedy)} = O(n\cdot r)$, where r is the rank of the matroid. Using Greedy, we are able to design its extended version ext-Greedy, formally outlined in Algorithm 2.

Algorithm 2 (Extended Greedy Algorithm for Submodular Optimization ext-Greedy)

Input: $\ell \geq 1$, monotone submodular function $g: 2^V \to \mathbb{R}_+$, matroid $\mathcal{M} = (V, \mathcal{I})$.
Output: sets $S_1, \ldots, S_\ell \in \mathcal{I}$.
1: **for** $\tau = 1, \ldots, \ell$, **do**
2: $S_\tau \leftarrow \emptyset$.
3: **while** S_τ is not a basis of $\mathcal{M}$, **do**
4: **Compute** $e^* = \mathrm{argmax}_{S_\tau + e \in \mathcal{I}}\{g(\cup_{j=1}^{\tau} S_j + e)\}$.
5: **Update** $S_\tau \leftarrow S_\tau + e^*$.

From Theorem 1, we can easily derive the following corollary for ext-Greedy.

Corollary 2. *For the offline robust submodular optimization problem (1), for any $0 < \epsilon, \delta < 1$, there is a polynomial-time bicriteria algorithm that uses ext-Greedy as a subroutine, runs in*

$$O\left(n\cdot r\cdot\log_2\left(\frac{k}{\epsilon}\right)\cdot\log(n)\cdot\min\left\{\frac{nk}{\epsilon}, \log_{1+\epsilon}\left(\max_{e,j} f_j(e)\right)\right\}\right)$$

time, and returns a set S^{ALG} such that

$$\min_{i\in[k]} f_i(S^{\mathrm{ALG}}) \geq (1-\epsilon)\cdot\max_{S\in\mathcal{I}}\min_{j\in[k]} f_j(S),$$

where $S^{\mathrm{ALG}} = S_1 \cup \cdots \cup S_\ell$, with $\ell = \lceil\log_2\frac{k}{\epsilon}\rceil$ and $S_1, \ldots, S_\ell \in \mathcal{I}$.

Another natural candidate is the standard local search algorithm, hereafter referred to as *LS* (Fisher et al. 1978). Roughly speaking, this algorithm starts with a maximal feasible set and iteratively swaps elements if the objective is improved while maintaining feasibility. If the objective cannot be improved, the algorithm stops. Fisher et al. (1978) prove that this procedure also achieves a 1/2-approximation, that is, $\beta = 1/2$. However, the running time of LS cannot be explicitly obtained. Finally, for the offline robust problem, the extended version of the local search algorithm ext-LS achieves the same guarantees as ext-Greedy but without an explicit running time.

2.1.3. Improving Running Time: Extended-Threshold Greedy. As mentioned earlier, we are interested in designing efficient bicriteria algorithms for the robust submodular problem (1). Unfortunately, the subroutine ext-Greedy performs $O(n\cdot r\cdot\log_2\frac{k}{\epsilon})$ function calls, which can be considerably inefficient when r is sufficiently large. Our objective in this section is to study a variant of the standard greedy algorithm that performs fewer function calls and whose running time does not depend on the rank of the matroid. For

our purposes, we consider ThGreedy, introduced by Badanidiyuru and Vondrák (2014). Roughly speaking, this procedure iteratively adds elements whose marginal value is above a certain threshold while maintaining feasibility. Unlike Greedy, ThGreedy may add more than one element in a single iteration. For the cardinality constraint, Badanidiyuru and Vondrák (2014) show that the ThGreedy algorithm achieves a $(1-1/e-\delta)$-approximation factor, where δ is the parameter that controls the threshold. Moreover, time(ThGreedy) $= O(\frac{n}{\delta}\log\frac{n}{\delta})$, which does not depend on the rank of the matroid. We formalize its extended version, ext-ThGreedy, in Algorithm 3. The original version corresponds to considering $\ell = 1$.

Algorithm 3 (Extended Threshold Greedy ext-ThGreedy)
Input: $\ell \geq 1$, ground set V with $n := |V|$, monotone submodular function $g : 2^V \to \mathbb{R}_+$, matroid $\mathcal{M} = (V, \mathcal{I})$, and $\delta > 0$.
Output: Feasible sets $S_1, \ldots, S_\ell \in \mathcal{I}$.
1: **for** $\tau = 1, \ldots, \ell$, **do**
2: $S_\tau \leftarrow \emptyset$.
3: $d \leftarrow \max_{e\in V} g(\cup_{j=1}^{\tau-1} S_j + e)$.
4: **for** $(\omega = d; \omega \geq \frac{\delta}{n} d; \omega \leftarrow (1-\delta)\omega)$ **do**
5: **for** $e \in V \backslash S_\tau$, **do**
6: **if** $S_\tau + e \in \mathcal{I}$ and $g_{\cup_{j=1}^{\tau} S_j}(e) \geq \omega$, **then**
7: $S_\tau \leftarrow S_\tau + e$.

Similarly to Badanidiyuru and Vondrák (2014), we can prove the following guarantee of ThGreedy when it is used for the problem of maximizing a single monotone submodular function subject to a matroid constraint.

Corollary 3. *For any $\delta > 0$, ThGreedy achieves a $(1 - \frac{1}{2-\delta})$-approximation for the problem of maximizing a single monotone submodular function subject to a matroid constraint, using $O(\frac{n}{\delta} \cdot \log\frac{n}{\delta})$ queries.*

For a detailed proof of Corollary 3, we refer the interested reader to the online supplement. Given the preceding result, we easily obtain Corollary 1 using $\beta = \frac{1}{2-\delta}$ in Theorem 1. The most relevant feature of ext-ThGreedy is its running time, which is independent on the rank of the matroid.

2.1.4. Tight Bounds: Extended Continuous Greedy.

To achieve a tight bound on the number of feasible sets in Theorem 1, we need to make use of the continuous greedy algorithm (Vondrák 2008), hereafter referred to as *CGreedy*. Before explaining the algorithm, let us recall some preliminary definitions. We denote the indicator vector of a set $S \subseteq V$ by $\mathbf{1}_S \in \{0,1\}^V$, where $\mathbf{1}_S(e) = 1$ if $e \in S$ and zero otherwise, and the matroid polytope by $\mathcal{P}(\mathcal{M}) = \text{conv}\{\mathbf{1}_S \mid S \in \mathcal{I}\}$. For any nonnegative set function $g : 2^V \to \mathbb{R}_+$, its multilinear extension $G : [0,1]^V \to \mathbb{R}_+$ is defined for any $y \in [0,1]^V$ as the expected value of $g(S_y)$, where S_y is the random set generated by drawing independently each element $e \in V$ with probability y_e. Formally,

$$G(y) = \mathbb{E}_{S\sim y}[g(S)] = \sum_{S\subseteq V} g(S) \prod_{e\in S} y_e \prod_{e\notin S} (1-y_e). \tag{4}$$

Observe that this is in fact an extension of g because for any subset $S \subseteq V$, we have $g(S) = G(\mathbf{1}_S)$. For any $x, y \in [0,1]^V$, we will denote $x \vee y$ as the vector whose components are $[x \vee y]_e = \max\{x_e, y_e\}$.

Fact 1 (Calinescu et al. 2011). Let g be a monotone submodular function and G its multilinear extension:

a. By the monotonicity of g, we have $\frac{\partial G}{\partial y_e} \geq 0$ for any $e \in V$. This implies that for any $x \leq y$ coordinate-wise, $G(x) \leq G(y)$. By contrast, by the submodularity of g, G is concave in any positive direction; that is, for any $e_1, e_2 \in V$, we have $\frac{\partial^2 G}{\partial y_{e_1} \partial y_{e_2}} \leq 0$.

b. Throughout the rest of this paper we will denote the gradient of G by $\nabla_e G(y) := \frac{\partial G}{\partial y_e}$, and the expected value of the marginal $g_S(e)$

$$\Delta_e G(y) := \mathbb{E}_{S\sim y}[g_S(e)]. \tag{5}$$

It is easy to see that $\Delta_e G(y) = (1-y_e)\nabla_e G(y)$. For any $x, y \in [0,1]^V$, it is easy to prove by using submodularity that

$$G(x \vee y) \leq G(x) + \Delta G(x) \cdot y \leq G(x) + \nabla G(x) \cdot y. \tag{6}$$

Broadly speaking, CGreedy works as follows. The algorithm starts with the empty set $y_0 = 0$ and for every $t \in [0,1]$ continuously finds a feasible direction z that maximizes $\nabla G(y_t) \cdot z$ over $\mathcal{P}(\mathcal{M})$, where y_t is the current fractional point. Then, CGreedy updates y_t according to z. Finally, the algorithm outputs a feasible set by rounding y_1 according to pipage rounding (Ageev and Sviridenko 2004), randomized pipage rounding, or randomized swap rounding (Chekuri et al. 2010). All these rounding procedures satisfy the following property: If S is the result of rounding y_1, then $S \in \mathcal{I}$ and $\mathbb{E}[g(S)] \geq G(y_1)$, where the expectation is taken over any randomization.

Notably, Vondrák (2008) proved that CGreedy finds a feasible set S such that $\mathbb{E}[g(S)] \geq (1-1/e) \cdot \max_{A\in\mathcal{I}} g(A)$. Unfortunately, time(CGreedy) $= O(n^8)$ because of the large number of samples required to accurately evaluate the multilinear extension. This running time can be substantially improved by using an accelerated version of the continuous greedy (ACGreedy) algorithm introduced by Badanidiyuru and Vondrák (2014). ACGreedy generalizes the idea of ThGreedy to the continuous framework and outputs a random feasible set S such that $\mathbb{E}[g(S)] \geq (1-1/e-\delta) \cdot \max_{A\in\mathcal{I}} g(A)$, where δ is the parameter that controls the threshold. More important, ACGreedy runs

in $O(rn\delta^{-4}\log^2 n)$, where r is the rank of the matroid. Therefore, by using ext-ACGreedy with $\beta = 1/e + \delta$ in Theorem 1, we obtain the following corollary.

Corollary 4. *For the offline robust submodular optimization problem (1), for any $0 < \epsilon < 1$, there is a polynomial-time bicriteria algorithm that uses ext-ACGreedy as a subroutine, runs in*

$$O\left(r \cdot n \cdot \delta^{-4} \cdot \log^2(n) \cdot \ln\left(\frac{k}{\epsilon}\right) \cdot \log(n) \cdot \min\left\{\frac{nk}{\epsilon}, \log_{1+\epsilon}\left(\max_{e,j} f_j(e)\right)\right\}\right)$$

time, and returns a set S^{ALG} such that

$$\mathbb{E}\left[\min_{i\in[k]} f_i\left(S^{\mathrm{ALG}}\right)\right] \geq (1-\epsilon) \cdot \max_{S\in\mathcal{I}} \min_{j\in[k]} f_j(S),$$

where $S^{\mathrm{ALG}} = S_1 \cup \cdots \cup S_\ell$, with $\ell = \lceil \ln \frac{k}{\epsilon} \rceil$ and $S_1, \ldots, S_\ell \in \mathcal{I}$.

As we can see, the number of independent sets required for obtaining this result $\ell = \lceil \ln \frac{k}{\epsilon} \rceil$ is smaller up to a constant than the number of sets obtained by ext-Greedy, $\ell = \lceil \log_2 \frac{k}{\epsilon} \rceil$. More important, Corollary 4 matches the hardness results given by Krause et al. (2008b).

2.2. Necessity of Monotonicity

In light of the approximation algorithms for nonmonotone submodular function maximization under matroid constraints (see, e.g., Lee et al. 2009), one might hope that an analogous bicriteria approximation algorithm could exist for robust nonmonotone submodular function maximization. However, we show that even without any matroid constraints, getting any approximation in the nonmonotone case is NP hard.

Lemma 1. *Unless P = NP, no polynomial-time algorithm can output a set $\tilde{S} \subseteq V$ given general submodular functions $f_1, \ldots, f_k$ such that $\min_{i\in[k]} f_i(\tilde{S})$ is within a positive factor of $\max_{S\subseteq V} \min_{i\in[k]} f_i(S)$.*

We use a reduction from SAT. Suppose that we have a SAT instance with variables $x_1, \ldots, x_n$. Consider $V = \{1, \ldots, n\}$. For every clause in the SAT instance, we introduce a nonnegative linear (and therefore submodular) function. For a clause $\bigvee_{i\in A} x_i \vee \bigvee_{i\in B} \overline{x_i}$, define the following:

$$f(S) := |S \cap A| + |B \setminus S|.$$

It is easy to see that f is linear and nonnegative. If we let S be the set of true variables in a truth assignment, then it is easy to see that $f(S) > 0$ if and only if the corresponding clause is satisfied. Consequently, finding a set S such that all functions f corresponding to different clauses are positive is as hard as finding a satisfying assignment for the SAT instance.

2.3. Computational Study

Our objective in this section is to empirically demonstrate that natural candidate subroutines for the offline robust problem (1) such as ext-Greedy can be substantially improved. For this, we will make use of ext-ThGreedy and other implementation improvements such as lazy evaluations and an early stopping rule. Ultimately, with these tools, we are able to design an efficient bicriteria algorithm.

First, let us recall how the general bicriteria algorithm works. In an outer loop we obtain an estimate γ of the value of the optimal solution $\mathrm{OPT} := \max_{S\in\mathcal{I}} \min_{i\in[k]} f_i(S)$ via a binary search. Next, for each guess γ, we define a new submodular set function as $g(S) := \frac{1}{k}\sum_{i\in[k]} \min\{f_i(S), \gamma\}$. Finally, we run Algorithm 1 to obtain a candidate solution. Depending on this result, we update the binary search on γ, and we iterate. We stop the binary search whenever we get a relative error of $1 - \epsilon/2$, namely $(1-\epsilon/2)\mathrm{OPT} \leq \gamma \leq \mathrm{OPT}$.

For our experimental study, we took into consideration the following:

1. *Reducing the number of function calls.* In Section 2.1.3, we theoretically showed that by using the subroutine ext-ThGreedy, we can get a (nearly) optimal objective value using fewer function evaluations than by using ext-Greedy at the cost of producing a slightly bigger family of feasible sets. Therefore, we will use ext-ThGreedy for our efficient bicriteria algorithm.

2. *Certifying (near) optimality.* The main bottleneck of any bicriteria algorithm remains obtaining a certificate of (near) optimality or, equivalently, a good upper bound on the optimum. We obtain that the optimum value is at most γ whenever running ext-$\mathcal{A}$ on function g fails to return a solution of desired objective. Because of the desired accuracy in binary search and the number of steps in the extended algorithm, obtaining good upper bounds on the optimum is computationally prohibitive. We resolve this issue by implementing an early stopping rule in the bicriteria algorithm. When running ext-$\mathcal{A}$ on function g (as explained earlier), we use the stronger guarantee given in Proposition 3: for any iteration $\tau \in [\ell]$, we obtain a set S_τ such that $g(\cup_{j=1}^{\tau} S_j) \geq (1-\beta^\tau)\cdot\gamma$. If in some iteration $\tau \in [\ell]$ the algorithm does not satisfy this guarantee, then it means that γ is much larger than OPT. In such a case, we stop ext-$\mathcal{A}$ and update the upper bound on OPT to be γ. This allows us to stop the iteration much earlier because in many real instances τ is typically much smaller than ℓ when γ is large. This leads to a drastic improvement in the number of function calls as well as central processing unit time.

3. *Lazy evaluations.* All greedy-like algorithms and baselines implemented in this section make use of lazy evaluations (Minoux 1978). This means that we keep a list of an upper bound $\rho(e)$ on the marginal gain for each element (initially ∞) in decreasing order, and at each iteration, it evaluates the element at the top of the list e'. If the marginal gain of this element satisfies $g_S(e') \geq \rho(e)$ for all $e \neq e'$, then submodularity ensures that $g_S(e') \geq g_S(e)$. In this way, for example, Greedy does not have to evaluate all marginal values to select the best element.

4. *Bounds initialization.* To compute the initial lower bound (LB) and upper bound (UB) for the binary search, we run the lazy greedy algorithm (Minoux 1978) for each function in a small subcollection $\{f_i\}_{i\in[k']}$, where $k' \ll k$, leading to k' solutions $A^1, \ldots, A^{k'}$ with guarantees $f_i(A^i) \geq (1/2) \cdot \max_{S\in\mathcal{I}} f_i(S)$. Therefore, we set $\text{UB} = 2 \cdot \min_{i\in[k']} f_i(A^i)$ and $\text{LB} = \max_{j\in[k']} \min_{i\in[k]} f_i(A^j)$. These two values correspond to the upper and lower bounds for the true optimum OPT.

To facilitate the interpretation of our theoretical results, we will consider partition constraints in all experiments: The ground set V is partitioned in q sets $\{P_1, \ldots, P_q\}$, and the family of feasible sets is $\mathcal{I} = \{S : |S \cap P_j| \leq b, \forall j \in [q]\}$, with the same budget b for each part. We test five methods: prev-extG, the extended greedy algorithm with no improvements, and the rest with improvements: ext-Greedy, ext-ThGreedy, and ext-SGreedy. The last method is a heuristic that uses the stochastic greedy algorithm (Mirzasoleiman et al. 2015) adapted to partition constraints (see the online supplement). The vanilla version of this algorithm samples a smaller ground set in each iteration and optimizes accordingly. Also, we tested the extended version of local search, but because of its bad performance compared with greedy-like algorithms, we do not report the results. For the final pseudocode of the main algorithm, we refer the interested reader to the online supplement.

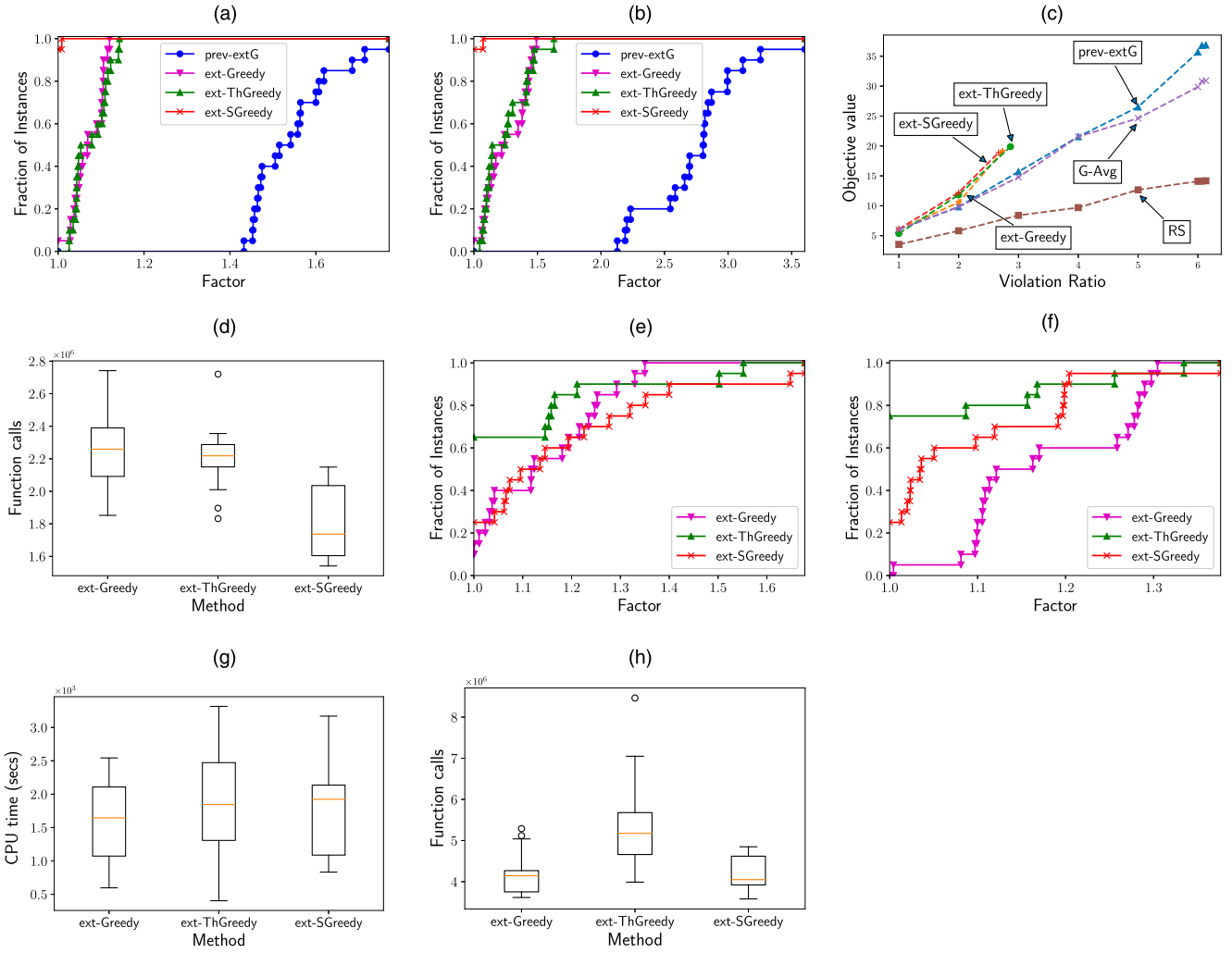
After running the four algorithms, we save the solution S^{ALG} with the largest violation ratio $\nu = \max_{j\in[q]} \lceil |S \cap P_j|/b \rceil$ and denote it by $\tau_{\max} := \lceil \nu \rceil$. Observe that $S^{\text{ALG}} = S_1 \cup \ldots \cup S_{\tau_{\max}}$, where $S_\tau \in \mathcal{I}$ for all $\tau \in [\tau_{\max}]$. We consider two additional baseline algorithms (without binary search): random selection, which outputs a set $\tilde{S} = \tilde{S}_1 \cup \ldots \cup \tilde{S}_{\tau_{\max}}$ such that for each $\tau \in [\tau_{\max}]$, $\tilde{S}_\tau$ is feasible, constructed by selecting elements uniformly at random, and $|\tilde{S}_\tau \cap P_j| = |S_\tau \cap P_j|$ for each part $j \in [q]$. Second, we run $\tau_{\max}$ times the lazy greedy algorithm on the average function $\frac{1}{k}\sum_{i\in[k]} f_i$ and consider constraints $\mathcal{I}_\tau = \{S : |S \cap P_j| \leq |S_\tau \cap P_j|, \forall j \in [q]\}$ for each iteration of $\tau \in [\tau_{\max}]$; we call this procedure G-Avg.

In all experiments, we consider the following parameters: approximation $1 - \epsilon = 0.99$, threshold $\delta = 0.1$, and sampling in ext-SGreedy with $\epsilon' = 0.1$. The composition of each part P_j is always uniformly at random from V.

2.3.1. Nonparametric Learning. We follow the setup in Mirzasoleiman et al. (2015). Let X_V be a set of random variables corresponding to biomedical measurements, indexed by a ground set of patients V. We assume X_V to be a Gaussian process (GP); that is, for every subset $S \subseteq V$, X_S is distributed according to a multivariate normal distribution $\mathcal{N}(\mu_S, \mathbf{\Sigma}_{S,S})$, where $\mu_S = (\mu_e)_{e\in S}$ and $\mathbf{\Sigma}_{S,S} = [\mathcal{K}_{e,e'}]_{e,e'\in S}$ are the prior mean vector and prior covariance matrix, respectively. The covariance matrix is given in terms of a positive-definite kernel $\mathcal{K}$; for example, a common choice in practice is the squared exponential kernel $\mathcal{K}_{e,e'} = \exp(-\|x_e - x_{e'}\|_2^2/h)$. Most efficient approaches for making predictions in GPs rely on choosing a small subset of data points. For instance, in the informative vector machine, the goal is to obtain a subset A that maximizes the information gain $f(A) = \frac{1}{2}\log\det(\mathbf{I} + \sigma^{-2}\mathbf{\Sigma}_{A,A})$, which is known to be monotone and submodular (Krause and Guestrin 2005). In our experiment, we use the Parkinson's telemonitoring data set (Tsanas et al. 2010) consisting of $n = 5{,}875$ patients with early-stage Parkinson's disease and the corresponding biomedical voice measurements with 22 attributes (dimension of the observations). We normalize the vectors to zero mean and unit norm. With these measurements, we computed the covariance matrix $\mathbf{\Sigma}$ considering the squared exponential kernel with parameter $h = 0.75$. For our robust criteria, we consider $k = 20$ perturbed versions of the information gain defined with $\sigma^2 = 1$; that is, Problem (1) corresponds to $\max_{A\in\mathcal{I}} \min_{i\in[20]} f(A) + \sum_{e\in A\cap\Lambda_i} \eta_e$, where $f(A) = \frac{1}{2}\log\det(\mathbf{I} + \mathbf{\Sigma}_{AA})$, Λ_i is a random set of size 1,000 with different composition for each $i \in [20]$, and $\eta \sim [0,1]^V$ is a uniform error vector.

We made 20 random runs considering $q = 3$ parts and budget $b = 5$. We report the results in Figure 1, (a)–(d). In Figure 1, (a) and (b), we show the performance profiles for the running time and the number of function calls of all methods, respectively. The y-axis corresponds to the fraction of the instances in which a specific method performs less than a multiplicative factor (x-axis) with respect to the best performance. For example, we observe in Figure 1(b) that in only 20% of the instances, the method prev-extG uses fewer than 2.5 times the number of function calls used by the best method. We also note that any of the three algorithms clearly outperform prev-extG either in terms of running time (Figure 1(a)) or function calls (Figure 1(b)). With this, we show empirically that our implementation improvements help the algorithm's

Figure 1. (Color online) Experimental Results



Notes: Nonparametric learning: performance profiles for (a) running time (to facilitate the visualization we compare the logarithm of the running times; note that ext-Greedy is covered by ext-ThGreedy) and (b) function calls. Box plot (c) shows the objective value versus the violation ratio in a single run of each method. Box plot (d) shows the function calls. Clustering: (small) performance profiles for (e) the running time and (f) the function calls, and (large) box plots for (g) the running time and (h) the function calls.

performance. We also note that ext-SGreedy is likely to have the best performance. Box plots for the function calls in Figure 1(d) confirm this fact because ext-SGreedy has the lowest median. Each method is presented in the x-axis, and the y-axis corresponds to the number of function calls (the line inside the box corresponds to the median, and circles are outliers). In this figure, we do not present the results of prev-extG because of the difference in magnitude. Finally, in Figure 1(c), we present the objective values (y-axis) obtained in a single run with respect to the number of feasible sets needed to cover the given set (x-axis). For example, when the set constructed by each method is two times the size of a feasible set, most of the procedures have an objective value around 10. We observe that the stopping rule is useful because the three tested algorithms output a nearly optimal solution earlier (using around three times the size of a feasible set), outperforming prev-extG and the benchmarks (which need around six times) and, moreover, at much less computational cost, as we mentioned earlier.

2.3.2. Exemplar-Based Clustering. We follow the setup in Mirzasoleiman et al. (2015). Solving the k-medoid problem is a common way to select a subset of exemplars that represent a large data set V (Kaufman and Rousseeuw 1990). This is done by minimizing the sum of pairwise dissimilarities between elements in $A \subseteq V$ and V. Formally, define $L(A) = \frac{1}{|V|}\sum_{e\in V} \min_{v\in A} d(e,v)$, where $d: V \times V \to \mathbb{R}_+$ is a distance function that represents the dissimilarity between a pair of elements. By introducing an appropriate

auxiliary element e_0, it is possible to define a new objective $f(A) := L(\{e_0\}) - L(A + e_0)$ that is monotone and submodular (Gomes and Krause 2010); thus, maximizing f is equivalent to minimizing L. In our experiment, we use the VOC2012 data set (Everingham et al. 2012). The ground set V corresponds to images, and we want to select a subset of the images that best represents the data set. Each image has several (possibly repeated) associated categories such as person and plane. There are around 20 categories in total. Therefore, images are represented by feature vectors obtained by counting the number of elements that belong to each category; for example, if an image has two people and one plane, then its feature vector is $(2, 1, 0, \ldots, 0)$ (where zeros correspond to other elements). We choose the Euclidean distance $d(e, e') = \|x_e - x_{e'}\|$, where $x_e, x_{e'}$ are the feature vectors for images e, e'. We normalize the feature vectors to mean zero and unit norm, and we choose e_0 as the origin. For our robust criteria, we consider $k = 20$ perturbations of the function f defined earlier; that is, Problem (1) corresponds to $\max_{A\in\mathcal{I}} \min_{i\in[20]} f(A) + \sum_{e\in A\cap\Lambda_i} \eta_e$, where Λ_i is a random set of fixed size with different composition for each $i \in [20]$, and finally, $\eta \sim [0, 1]^V$ is a uniform error vector.

We consider two experiments: (small) with $n =$ 3,000 images, 20 random instances considering $q = 6$ parts, and budget $b = 70$, $|\Lambda_i| = 500$; and (large) with $n = 17{,}125$ images, 20 random instances considering $q \in \{10, \ldots, 29\}$ parts, and budget $b = 5$, $|\Lambda_i| = 3{,}000$ (we do not implement prev-extG because of the exorbitant running time). We report the results of the experiments in Figure 1, (e)–(h). In Figure 1, (e) and (f), we show the performance profiles for the running time and the number of function calls of all methods, respectively. The y-axis corresponds to the fraction of the instances in which a specific method performs less than a multiplicative factor (x-axis) with respect to the best method. For example, we observe in Figure 1(f) that in only 20% of the instances, the method ext-Greedy uses fewer than 1.1 times the number of function calls used by the best method (ext-ThGreedy). We do not present the results of prev-extG because of its difference in magnitude. For the small experiments, the performance profiles in Figure 1, (e) and (f), confirm our theoretical results: ext-ThGreedy is the most likely to use fewer function calls (Figure 1(f)) and less running time (Figure 1(e)) when the rank is relatively high (in this case, $q \cdot b = 420$). This contrasts with the performance of ext-Greedy that depends on the rank (the chart in Figure 1(f) reflects this). For large experiments, we can see in the box plots in Figure 1, (g) and (h), that the results are similar in terms of either running time or function evaluations. Therefore, when we face a large ground set and a small rank, we could choose any algorithm, but we would still prefer ext-ThGreedy because it has no dependency on the rank.

3. The Online Case

In this section, we consider the online robust optimization problem (Equation (2)) under matroid constraints. We introduce an online bicriteria algorithm that achieves a sublinear $(1 - \epsilon)$-regret while using solution S^t at time t that is a union of $O(\ln \frac{1}{\epsilon})$ independent sets from $\mathcal{I}$. To start, let us first present definitions and known results that play a key role in this online optimization problem. To avoid any confusion, in the remainder of this section we will denote the dot product between two vectors as $\langle \cdot, \cdot \rangle$.

3.1. Background

3.1.1. Submodular Maximization. Multilinear extension plays a crucial role in designing approximation algorithms for various constrained submodular optimization problems (see Section 2.1.4 for a list of its useful properties). Vondrák (2008) introduced the discretized continuous greedy algorithm that achieves a $1 - 1/e$ approximate solution for maximizing a single monotone submodular function under matroid constraints (see Feldman et al. 2011 for the variant of the continuous greedy that we use). Consider G to be the multilinear extension of a monotone submodular function g. Recall that $\Delta G(y)$ denotes the vector whose eth coordinate is $\Delta_e G(y)$, as defined in (5). At a high level, the discretized continuous greedy algorithm discretizes interval $[0, 1]$ into points $\{0, \delta, 2\delta, \ldots, 1\}$. Starting at $y_0 = 0$, for each $\tau \in \{\delta, 2\delta, \ldots, 1\}$, the algorithm uses a linear program to compute the direction $z_\tau = \arg\max_{z\in\mathcal{P}(\mathcal{M})} \langle \Delta G(y_{\tau-\delta}), z \rangle$. Then the algorithm takes a step in the direction of z_τ by setting $y_{\tau,e} \leftarrow y_{\tau-\delta,e} + \delta z_{\tau,e}(1 - y_{\tau-\delta,e})$ for all $e \in V$. Finally, it outputs a set S by rounding the fractional solution y_1. We will use this discretized version of the continuous greedy algorithm to construct our online algorithm in the next section.

3.1.2. The Soft-Min Function. Consider a set of k twice-differentiable real-valued functions $\phi_1, \ldots, \phi_k : \mathbb{R}^n \to \mathbb{R}$. Let $\phi_{\min}$ be the minimum among these functions; that is, for each point x in the domain, define $\phi_{\min}(x) := \min_{i\in[k]} \phi_i(x)$. This function can be approximated by using the so-called soft-min (or log-sum-exp) function $H : \mathbb{R}^n \to \mathbb{R}$, defined as follows:

$$H(x) = -\frac{1}{\alpha} \ln \sum_{i\in[k]} e^{-\alpha\phi_i(x)},$$

where $\alpha > 0$ is a fixed parameter. We now present some of the key properties of this function in the following lemma.

Lemma 2. *For any set of k twice-differentiable real-valued functions $\phi_1, \dots, \phi_k$, the soft-min function H satisfies the following properties:*

a. *Bounds:*

$$\phi_{\min}(x) - \frac{\ln k}{\alpha} \le H(x) \le \phi_{\min}(x). \tag{7}$$

b. *Gradient:*

$$\nabla H(x) = \sum_{i\in[k]} p_i(x)\nabla\phi_i(x), \tag{8}$$

where $p_i(x) := e^{-\alpha\phi_i(x)} / \sum_{j\in[k]} e^{-\alpha\phi_j(x)}$. Clearly, if $\nabla\phi_i \ge 0$ for all $i \in [k]$, then $\nabla H \ge 0$.

c. *Hessian:*

$$\frac{\partial^2 H(x)}{\partial x_{e_1}\partial x_{e_2}} = \sum_{i\in[k]} p_i(x)\left(-\alpha\frac{\partial\phi_i(x)}{\partial x_{e_1}}\frac{\partial\phi_i(x)}{\partial x_{e_2}} + \frac{\partial^2\phi_i(x)}{\partial x_{e_1}\partial x_{e_2}}\right) + \alpha\nabla_{e_1}H(x)\nabla_{e_2}H(x). \tag{9}$$

Moreover, if for all $i \in [k]$, we have $|\frac{\partial\phi_i}{\partial x_{e_1}}| \le L_1$ and $|\frac{\partial^2\phi_i}{\partial x_{e_1}\partial x_{e_2}} s| \le L_2$, then $|\frac{\partial^2 H}{\partial x_{e_1}\partial x_{e_2}}| \le 2\alpha L_1^2 + L_2$.

c. *Comparing the average of the ϕ_i functions with H. Given $\alpha > 0$, we have*

$$H(x) \le \sum_{i\in[k]} p_i(x)\phi_i(x) \le H(x) + \frac{\ln\alpha}{\alpha} + \frac{\ln k}{\alpha} + \frac{k}{\alpha}. \tag{10}$$

Therefore, for $\alpha > 0$ sufficiently large, $\sum_{i\in[k]} p_i(x)\phi_i(x)$ is a good approximation of $H(x)$.

For other properties and applications, we refer the interested reader to (Calafiore and El Ghaoui 2014). Now, we present a lemma that is used to prove the main result in the online case, Theorem 2.

Lemma 3. *Fix a parameter $\delta > 0$. Consider T collections of k twice-differentiable functions, namely $\{\phi_i^1\}_{i\in[k]}, \dots, \{\phi_i^T\}_{i\in[k]}$. Assume that $0 \le \phi_i^t(x) \le 1$ for any x in the domain, for all $t \in [T]$ and $i \in [k]$. Define the corresponding sequence of soft-min functions $H^1, \dots, H^T$ with common parameter $\alpha > 0$. Then any two sequences of points $\{x^t\}_{t\in[T]}, \{y^t\}_{t\in[T]} \subseteq [0,1]^V$ with $|x^t - y^t| \le \delta$ satisfy*

$$\sum_{t\in[T]} H^t(y^t) - \sum_{t\in[T]} H^t(x^t) \ge \sum_{t\in[T]} \langle \nabla H^t(x^t), y^t - x^t\rangle - O(Tn^2\delta^2\alpha).$$

For a proof of these lemmas, we refer readers to the online supplement.

3.2. Online Algorithm and Analysis

Turning our attention to the online robust optimization problem (2), we are immediately faced with two challenges. First, we need to find a direction z_t that is good for all k submodular functions in an online fashion. In the offline case, we used the function $g(S) = \frac{1}{k}\cdot\sum_{i=1}^k \min\{f_i(S), \gamma\}$ and its multilinear extension to find such a direction (see Section 2.1.4). To resolve this issue, we use a soft-min function that converts robust optimization over k functions into optimizing of a single function. Second, robust optimization leads to nonconvex and nonsmooth optimization combined with online arrival of such submodular functions. To deal with this, we use the FPL online algorithm introduced by Kalai and Vempala (2005).

For any collection of monotone submodular functions $\{f_i^t\}_{i\in[k]}$ played by the adversary, we define the soft-min function with respect to the corresponding multilinear extensions $\{F_i^t\}_{i\in[k]}$ as $H^t(y) := -\frac{1}{\alpha}\ln \sum_{i\in[k]} e^{-\alpha F_i^t(y)}$, where $\alpha > 0$ is a suitable parameter. Recall that if we assume functions f_i^t taking values in $[0,1]$, then their multilinear extensions F_i^t also take values in $[0,1]$. The following properties of the soft-min function as defined in the preceding section are easy to verify and crucial for our result:

1. *Approximation:* $\min_{i\in[k]} F_i^t(y) - \frac{\ln k}{\alpha} \le H^t(y) \le \min_{i\in[k]} F_i^t(y)$.

2. *Gradient:* $\nabla H^t(y) = \sum_{i\in[k]} p_i^t(y)\nabla F_i^t(y)$, where $p_i^t(y) \propto e^{-\alpha F_i^t(y)}$ for all $i \in [k]$.

Note that as α increases, the soft-min function H^t becomes a better approximation of $\min_{i\in[k]}\{F_i^t\}$; however, its smoothness degrades (see property (9) in Section 3.1). By contrast, the second property shows that the gradient of the soft-min function is a convex combination of the gradients of the multilinear extensions, which allows us to optimize all the functions at the same time. Indeed, define $\Delta_e H^t(y) := \sum_{i\in[k]} p_i^t(y)\Delta_e F_i^t(y) = (1 - y_e)\nabla_e H^t(y)$. At each stage $t \in [T]$, we use the information from the gradients previously observed, in particular, $\{\Delta H^1, \cdots, \Delta H^{t-1}\}$, to decide the set S^t. To deal with adversarial input functions, we use the FPL algorithm (Kalai and Vempala 2005) and the following guarantee about the algorithm.

Theorem 4 (Kalai and Vempala 2005)**.** *Let $s_1, \dots, s_T \in \mathcal{S}$ be a sequence of rewards. The FPL Algorithm 6 (in the* online supplement*) with parameter $\eta \le 1$ outputs decisions $d_1, \dots, d_T$ with regret*

$$\max_{d\in\mathcal{D}} \sum_{t\in[T]} \langle s_t, d\rangle - \mathbb{E}\left[\sum_{t\in[T]} \langle s_t, d_t\rangle\right] \le O\left(poly(n)\left(\eta T + \frac{1}{T\eta}\right)\right).$$

For completeness, we include the original setup and the algorithm in the online supplement.

Our online algorithm works as follows. First, given $0 < \epsilon < 1$, we denote $\ell := \lceil \ln\frac{1}{\epsilon}\rceil$. We consider the following discretization indexed by $\tau \in \{0, \delta, 2\delta, \dots, \ell\}$ and construct fractional solutions y_τ^t for each iteration t and discretization index τ. At each iteration t, ideally, we would like to construct $\{y_\tau^t\}_{\tau=0}^{\ell}$ by running the continuous greedy algorithm using the soft-min

function H^t and then play S^t using these fractional solutions. In the online model, though, function H^t is revealed only after playing set S^t. To remedy this, we aim to construct y_τ^t using the FPL algorithm based on gradients $\{\nabla H^j\}_{j=1}^{t-1}$ obtained from previous iterations. Thus we have multiple FPL instances, one for each discretization parameter, being run by the algorithm. Finally, at the end of iteration t, we have a fractional vector y_ℓ^t, which belongs to $\ell \cdot \mathcal{P}(\mathcal{M}) \cap [0,1]^V$ and therefore can be written, fractionally, as a union of ℓ independent sets using the matroid union theorem (Schrijver 2003).

We round the fractional solution y_ℓ^t using the randomized swap rounding (or randomized pipage rounding) proposed by Chekuri et al. (2010) for matroid $\mathcal{M}_\ell$ to obtain the set S^t to be played at time t. The following theorem from Chekuri et al. (2010) gives the necessary property of the randomized swap rounding that we use.

Theorem 5 (Chekuri et al. 2010, theorem II.1)**.** *Let f be a monotone submodular function and F its multilinear extension. Let $x \in \mathcal{P}(\mathcal{M}')$ be a point in the polytope of matroid $\mathcal{M}'$ and S' a random independent set obtained from it by randomized swap rounding. Then $\mathbb{E}[f(S')] \geq F(x)$.*

We formalize the details in Algorithm 4 (observe that $\ell/\delta \in \mathbb{Z}_+$).

Algorithm 4 (OnlineSoftMin Algorithm)

Input: Learning parameter $\eta > 0$, $\epsilon > 0$, $\alpha = nT$, discretization $\delta = n^{-1}T^{-1}$, and $\ell = \lceil \ln \frac{1}{\epsilon} \rceil$.
Output: Sequence of sets $S_1, \ldots, S_T$.
1: Sample $q \sim [0, 1/\eta]^V$.
2: **for** $t = 1$ to T, **do**
3: $y_0^t = 0$,
4: **for** $\tau \in \{\delta, 2\delta, \ldots, \ell\}$, **do**
5: **Compute** $z_\tau^t = \arg\max_{z \in \mathcal{P}(\mathcal{M})} \langle \sum_{j=1}^{t-1} \Delta H^j(y_{\tau-\delta}^j) + q, z \rangle$.
6: **Update:** For each $e \in V$, $y_{\tau,e}^t = y_{\tau-\delta,e}^t + \delta(1 - y_{\tau-\delta,e}^t) z_{\tau,e}^t$.
7: **Play** $S^t \leftarrow$ Rounding(y_ℓ^t). **Receive** and **observe** new collection $\{f_i^t\}_{i \in [k]}$.

To get sublinear regret for the FPL Algorithm 6, Kalai and Vempala (2005) assume a couple of conditions on the problem (see the online supplement). Similarly, for our online model, we need to consider the following for any $t \in [T]$:

1. Bounded diameter of $\mathcal{P}(\mathcal{M})$; that is, for all $z, z' \in \mathcal{P}(\mathcal{M})$, $\|z - z'\|_1 \leq D$.
2. For all $y, z \in \mathcal{P}(\mathcal{M})$, we require $|\langle z, \Delta H^t(y) \rangle| \leq L$.
3. For all $y \in \mathcal{P}(\mathcal{M})$, we require $\|\Delta H^t(y)\|_1 \leq A$.

Now we give a complete proof of Theorem 2 for any given learning parameter $\eta > 0$, but the final result follows with $\eta = \sqrt{D/LAT}$ and assuming $L \leq n$, $A \leq n$, and $D \leq \sqrt{n}$, which gives a $O(n^{5/4})$ dependency on the dimension in the regret.

Proof of Theorem 2. Consider the sequence of multilinear extensions $\{F_i^1\}_{i \in [k]}, \ldots, \{F_i^T\}_{i \in [k]}$ derived from the monotone submodular functions f_i^t obtained during the dynamic process. Because the f_i^t's have value in $[0,1]$, we have $0 \leq F_i^t(y) \leq 1$ for any $y \in [0,1]^V$ and $i \in [k]$. Consider the corresponding soft-min functions H^t for collection $\{F_i^t\}_{i \in [k]}$ with $\alpha = nT$ for all $t \in [T]$. Denote $\ell = \lceil \ln \frac{1}{\epsilon} \rceil$ and fix $\tau \in \{\delta, 2\delta, \ldots, \ell\}$ with $\delta = n^{-1}T^{-1}$. According to the update in Algorithm 4, $\{y_\tau^t\}_{t \in [T]}$ and $\{y_{\tau-\delta}^t\}_{t \in [T]}$ satisfy conditions of Lemma 3. Thus, we obtain the following:

$$\sum_{t \in [T]} H^t(y_\tau^t) - H^t(y_{\tau-\delta}^t) \geq \sum_{t \in [T]} \langle \nabla H^t(y_{\tau-\delta}^t), y_\tau^t - y_{\tau-\delta}^t \rangle - O(Tn^2\delta^2\alpha).$$

Then, because the update is $y_{\tau,e}^t = y_{\tau-\delta,e}^t + \delta(1 - y_{\tau-\delta,e}^t) z_{\tau,e}^t$, we get the following:

$$\begin{aligned} &\sum_{t \in [T]} H^t(y_\tau^t) - H^t(y_{\tau-\delta}^t) \\ &\quad \geq \delta \sum_{t \in [T]} \sum_{e \in V} \nabla_e H^t(y_{\tau-\delta}^t)(1 - y_{\tau-\delta,e}^t) z_{\tau,e}^t \\ &\qquad - O\left(Tn^2\delta^2\alpha\right) \\ &\quad = \delta \sum_{t \in [T]} \langle \Delta H^t(y_{\tau-\delta}^t), z_\tau^t \rangle - O\left(Tn^2\delta^2\alpha\right). \end{aligned} \tag{11}$$

Observe that an FPL algorithm is implemented for each τ, so we can state a regret bound for each τ using Theorem 4. Specifically,

$$\mathbb{E}\left[\sum_{t \in [T]} \langle \Delta H^t(y_{\tau-\delta}^t), z_\tau^t \rangle\right] \geq \max_{z \in \mathcal{P}(\mathcal{M})} \mathbb{E}\left[\sum_{t \in [T]} \langle \Delta H^t(y_{\tau-\delta}^t), z \rangle\right] - R_\eta,$$

where $R_\eta = \eta LAT + \frac{D}{\eta}$ is the regret guarantee for a given $\eta > 0$. By taking expectation in (11) and using the regret bound we just mentioned, we obtain the following:

$$\begin{aligned} &\mathbb{E}\left[\sum_{t \in [T]} H^t(y_\tau^t) - H^t(y_{\tau-\delta}^t)\right] \\ &\quad \geq \delta\left(\max_{z \in \mathcal{P}(\mathcal{M})} \mathbb{E}\left[\sum_{t \in [T]} \langle \Delta H^t(y_{\tau-\delta}^t), z \rangle\right]\right) - \delta R_\eta \\ &\qquad - O(Tn^2\delta^2\alpha) \\ &\quad \geq \delta\mathbb{E}\left(\sum_{t \in [T]}\left[H^t(x^*) - \sum_{i \in [k]} p_i^t(y_{\tau-\delta}^t) F_i^t(y_{\tau-\delta}^t)\right]\right) \\ &\qquad - \delta R_\eta - O(Tn^2\delta^2\alpha), \end{aligned} \tag{12}$$

where $x^* = \mathbf{1}_{S^*}$ is the indicator vector of the true optimum S^* for $\max_{S \in \mathcal{I}} \sum_{t \in [T]} \min_{i \in [k]} f_i^t(S)$. Observe that (12)

follows from the monotonicity and submodularity of each f_i^t; specifically, we know that

$$\begin{aligned}\langle \Delta H^t(y), z\rangle &= \sum_{i\in[k]} p_i^t(y)\langle \Delta F_i^t(y), z\rangle \\ &\geq \sum_{i\in[k]} p_i^t(y)F_i^t(x^*) - \sum_{i\in[k]} p_i^t(y)F_i^t(y) \qquad \text{(Equation (6))} \\ &\geq F_{\min}^t(x^*) - \sum_{i\in[k]} p_i^t(y)F_i^t(y) \\ &\geq H^t(x^*) - \sum_{i\in[k]} p_i^t(y)F_i^t(y).\end{aligned}$$

By applying property (10) of the soft-min in expression (12), we get the following:

$$\begin{aligned}&\mathbb{E}\left[\sum_{t\in[T]} H^t(y_\tau^t) - H^t(y_{\tau-\delta}^t)\right] \\ &\quad\geq \delta\mathbb{E}\left(\sum_{t\in[T]} H^t(x^*) - H^t(y_{\tau-\delta}^t)\right) - \delta R_\eta \\ &\qquad - O(Tn^2\delta^2\alpha) \\ &\qquad - \delta T\left(\frac{\ln\alpha}{\alpha} + \frac{\ln k}{\alpha} + \frac{k}{\alpha}\right). \qquad (13)\end{aligned}$$

Given the choice of α and δ, the last two terms on the right-hand side of inequality (13) are small compared with R_η, so by rearranging terms, we can state the following:

$$\begin{aligned}&\sum_{t\in[T]} H^t(x^*) - \mathbb{E}\left[\sum_{t\in[T]} H^t(y_\tau^t)\right] \\ &\quad\leq (1-\delta)\left(\sum_{t\in[T]} H^t(x^*) - \mathbb{E}\left[\sum_{t\in[T]} H^t(y_{\tau-\delta}^t)\right]\right) \\ &\qquad + 2\delta R_\eta.\end{aligned}$$

By iterating $\frac{\ell}{\delta}$ times in τ, we get the following:

$$\begin{aligned}&\sum_{t\in[T]} H^t(x^*) - \mathbb{E}\left[\sum_{t\in[T]} H^t(y_\ell^t)\right] \\ &\quad\leq (1-\delta)^{\frac{\ell}{\delta}}\left(\sum_{t\in[T]} H^t(x^*) - \sum_{t\in[T]} H^t(y_0^t)\right) \\ &\qquad + O\left(\left(1-(1-\delta)^{\frac{\ell}{\delta}}\right)R_\eta\right) \\ &\quad\leq \epsilon\left[\sum_{t\in[T]} H^t(x^*) + \ln k\right] + O\big((1-\epsilon)R_\eta\big),\end{aligned}$$

where in the last inequality we used $(1-\delta)\leq e^{-\delta}$ and $\ell = \lceil \ln\frac{1}{\epsilon}\rceil$. Given that the term $\epsilon\ln k$ is small (for ϵ sufficiently small), we can bound it by $O(R_\eta)$. Because α is sufficiently large, we can apply the approximation property of soft-min functions to obtain the following regret bound:

$$\begin{aligned}&(1-\epsilon)\cdot\sum_{t\in[T]}\min_{i\in[k]} F_i^t(x^*) - \mathbb{E}\left[\sum_{t\in[T]}\min_{i\in[k]} F_i^t(y_\ell^t)\right] \\ &\quad\leq O\big((1-\epsilon)R_\eta\big).\end{aligned}$$

Because we are doing randomized swap rounding (or randomized pipage rounding) on each y_ℓ^t, Theorem 5 shows that there is a random set S^t that is independent in $\mathcal{M}_\ell$ (i.e., S^t is the union of at most ℓ independent sets in $\mathcal{I}$) such that $\mathbb{E}[f_i^t(S^t)]\geq F_i^t(y_\ell^t)$ for all $t\in[T]$ and $i\in[k]$. Thus, we finally obtain the following:

$$\begin{aligned}&(1-\epsilon)\cdot\max_{S\in\mathcal{I}}\sum_{t\in[T]}\min_{i\in[k]} f_i^t(S) - \sum_{t\in[T]}\min_{i\in[k]}\mathbb{E}\big[f_i^t(S^t)\big] \\ &\quad\leq O\big((1-\epsilon)R_\eta\big). \quad \square\end{aligned}$$

Observation 1. Theorem 2 could be easily extended to an adaptive adversary by sampling in each stage $t\in[T]$ a different perturbation $q_t\sim[0,1/\eta]^V$, as shown in Kalai and Vempala (2005).

Note that the guarantee of Theorem 2 holds with respect to the minimum of $\mathbb{E}[f_i^t(S^t)]$, as opposed to the guarantee of Theorem 1, which directly bounds the minimum of $f_i(S)$. Because of this, the online algorithm needs only $\lceil\ln\frac{1}{\epsilon}\rceil$ independent sets, compared with the offline solution, which needs $\lceil\log\frac{k}{\epsilon}\rceil$ independent sets. It might seem more appealing to define the regret with respect to the expected value of the minimum function, but at the same time, it becomes technically more challenging for several reasons. If one wants to apply a similar technique to the offline algorithm, it is not clear how to dynamically estimate the optimal value of the online model while considering simultaneously the behavior of the adversary and the nonsmoothness of the minimum function.

4. Extensions

In this section, we consider other classes of combinatorial constraints for the offline robust model. Because the extended algorithm ext-$\mathcal{A}$ considers a general algorithm $\mathcal{A}$ for submodular maximization, we can expand our results to other constraints such as knapsack constraints and multiple matroids. Similar results can be obtained in the online model as long as the polytope is downward closed.

4.1. Knapsack Constraint

Consider a knapsack constraint $\mathcal{K} = \{S\subseteq[n] : \sum_{e\in S} c_e\leq 1\}$, where $c_e > 0$ for all $e\in[n]$. Our interest is to solve the following robust problem:

$$\max_{S\in\mathcal{K}}\min_{i\in[k]} f_i(S). \qquad (14)$$

Corollary 5. *For Problem (14), there is a polynomial-time algorithm that returns a set S^{ALG} such that for all $i \in [k]$, for a given $0 < \epsilon < 1$,*

$$f_i(S^{\mathrm{ALG}}) \geq (1-\epsilon) \cdot \max_{S \in \mathcal{K}} \min_{j \in [k]} f_j(S)$$

and $\sum_{e \in S^{\mathrm{ALG}}} c_e \leq \ell$ for $\ell = O(\ln \frac{k}{\epsilon})$. Moreover, S^{ALG} can be covered by at most ℓ sets in $\mathcal{K}$.

Following the idea of the general extended algorithm ext-$\mathcal{A}$, we design an extended version of the bang-per-buck greedy algorithm. We formalize this procedure in Algorithm 5. Even though the standard bang-per-buck greedy algorithm does not provide any approximation factor, if we relax the knapsack constraint to be $\sum_{e \in S} c_e \leq 2$, then the algorithm gives a $1 - 1/e$ factor. There are other approaches to avoid this relaxation (see, e.g., Sviridenko 2004).

Algorithm 5 (Extended Bang-per-Buck Algorithm for Knapsack Constraints)

Input: $\ell \geq 1$, monotone submodular function $g : 2^V \to \mathbb{R}_+$, knapsack constraint $\mathcal{K}$.

Output: Sets $S_1, \ldots, S_\ell \in \mathcal{K}$.

1: **for** $\tau = 1, \ldots, \ell$, **do**
2: $S_\tau \leftarrow \emptyset$.
3: **while** $V \neq \emptyset$, **do**
4: **Compute** $e^* = \operatorname{argmax}_{e \in V} \{\frac{g(\cup_{j=1}^{\tau} S_j + e) - g(\cup_{j=1}^{\tau} S_j)}{c_e}\}$.
5: **if** $\sum_{e \in S^\tau} c_e + c_{e^*} \leq 2$, **then** $S_\tau \leftarrow S_\tau + e^*$.
6: $V \leftarrow V - e^*$.
7: **Restart** ground set V.

Given a monotone submodular function $g : 2^V \to \mathbb{R}_+$, Algorithm 5 produces a set $S^{\mathrm{ALG}} = S_1 \cup \cdots \cup S_\ell$ such that $g(S^{\mathrm{ALG}}) \geq (1 - \frac{1}{e^r}) \cdot \max_{S \in \mathcal{K}} g(S)$. Therefore, Corollary 5 can be easily proved by defining g in the same way as in Theorem 1 and running Algorithm 5 on g with $\ell = O(\ln \frac{k}{\epsilon})$.

4.2. Multiple Matroid Constraints

Consider a family of r matroids $\mathcal{M}_j = (V, \mathcal{I}_j)$ for $j \in [r]$. Our interest is to solve the following robust problem:

$$\max_{S \in \bigcap_{j=1}^{r} \mathcal{I}_j} \min_{i \in [k]} f_i(S). \tag{15}$$

Corollary 6. *For Problem (15), there is a polynomial-time algorithm that returns a set S^{ALG} such that for all $i \in [k]$, for a given $0 < \epsilon < 1$,*

$$f_i(S^{\mathrm{ALG}}) \geq (1-\epsilon) \cdot \max_{S \in \bigcap_{j=1}^{r} \mathcal{I}_j} \min_{i \in [k]} f_i(S),$$

where S^{ALG} is the union of $O(\log \frac{k}{\epsilon} / \log \frac{r+1}{r})$ independent sets in $\mathcal{I}$.

Fisher et al. (1978) proved that the standard greedy algorithm gives a $1/(1+r)$ approximation for Problem (15) when $k = 1$. Therefore, we can adapt Algorithm 2 to produce a set $S^{\mathrm{ALG}} = S_1 \cup \cdots \cup S_\ell$ such that

$$f(S^{\mathrm{ALG}}) \geq \left(1 - \left(\frac{r}{r+1}\right)^{\ell}\right) \cdot \max_{S \in \bigcap_{j=1}^{r} \mathcal{I}_j} f(S).$$

Then Corollary 6 can be proved similarly to Theorem 1 by choosing $\ell = O(\log \frac{k}{\epsilon} / \log \frac{r+1}{r})$.

4.3. Distributionally Robust Over Polyhedral Sets

Let $\mathcal{Q} \subseteq \Delta(k)$ be a polyhedral set, where $\Delta(k)$ is the probability simplex on k elements. For $q \in \mathcal{Q}$, denote $f_q := q_1 f_1 + \cdots + q_k f_k$, which is also monotone and submodular. Given a matroid $\mathcal{M} = (V, \mathcal{I})$, our interest is to solve the following distributionally robust problem:

$$\max_{S \in \mathcal{I}} \min_{q \in \mathcal{Q}} f_q(S). \tag{16}$$

Denote by $\mathrm{Vert}(\mathcal{Q})$ the set of extreme points of $\mathcal{Q}$, which is finite because $\mathcal{Q}$ is polyhedral. Then Problem (16) is equivalent to $\max_{S \in \mathcal{I}} \min_{q \in \mathrm{Vert}(\mathcal{Q})} f_q(S)$. Then we can easily derive Corollary 7 by applying Theorem 1 in the equivalent problem. Note that when $\mathcal{Q}$ is the simplex, we get the original Theorem 1.

Corollary 7. *For Problem (16), there is a polynomial-time algorithm that returns a set S^{ALG} such that for all $i \in [k]$, for a given $0 < \epsilon < 1$,*

$$f_i(S^{\mathrm{ALG}}) \geq (1-\epsilon) \cdot \max_{S \in \mathcal{I}} \min_{q \in \mathcal{Q}} f_q(S),$$

with $S^{\mathrm{ALG}} = S_1 \cup \cdots \cup S_\ell$ for $\ell = O(\log \frac{|\mathrm{Vert}(\mathcal{Q})|}{\epsilon})$ and $S_1, \ldots, S_\ell \in \mathcal{I}$.

Acknowledgments

The authors thank Shabbir Ahmed for discussions about the distributionally robust problem (16).

References

Ageev A, Sviridenko M (2004) Pipage rounding: A new method of constructing algorithms with proven performance guarantee. *J. Combin. Optim.* 8(3):307–328.

Anari N, Haghtalab N, Naor S, Pokutta S, Singh M, Torrico A (2019) Structured robust submodular maximization: Offline and online algorithms. Chaudhuri K, Sugiyama M, eds. *Proc. 22nd Internat. Conf. Artificial Intelligence Statist.* (AISTATS, Okinawa, Japan), 3128–3137.

Badanidiyuru A, Vondrák J (2014) Fast algorithms for maximizing submodular functions. Chaudhuri K, Sugiyama M, eds.*Proc. 25th Annual ACM-SIAM Symp. Discrete Algorithms* (SODA) (SIAM, Philadelphia), 1497–1514.

Bogunovic I, Mitrovic S, Scarlett J, Cevher V (2017) Robust submodular maximization: A non-uniform partitioning approach. Precup D, Teh YW, eds. *Proc. 34th Internat. Conf. Machine Learn.* (ICML, Sydney), 508–516.

Buchbinder N, Feldman M (2016) Deterministic algorithms for submodular maximization problems. *Proc. 27th Annual ACM-SIAM Symp. Discrete Algorithms* (SODA) (ACM, New York), 392–403.

Buchbinder N, Feldman M, Schwartz R (2016) Comparing apples and oranges: Query trade-off in submodular maximization. *Math. Oper. Res.* 42(2):308–329.

Buchbinder N, Feldman M, Naor JS, Schwartz R (2014) Submodular maximization with cardinality constraints. *Proc. 25th Annual ACM-SIAM Symp. Discrete Algorithms* (SODA) (ACM, New York), 1433–1452.

Calafiore GC, El Ghaoui L (2014) *Optimization Models* (Cambridge University Press, Cambridge, UK).

Calinescu G, Chekuri C, Pál M, Vondrák J (2011) Maximizing a monotone submodular function subject to a matroid constraint. *SIAM J. Comput.* 40(6):1740–1766.

Chekuri C, Vondrak J, Zenklusen R (2010) Dependent randomized rounding via exchange properties of combinatorial structures. *Proc. 51st Annual Symp. Foundations Comput. Sci.* (FOCS, Las Vegas, NV), 575–584.

Chen RS, Lucier B, Singer Y, Syrgkanis V (2017) Robust optimization for non-convex objectives. Guyon I, Luxburg UV, Bengio S, Wallach H, Fergus R, Vishwanathan S, Garnett R, eds. *Proc. 31st Internat. Conf. Neural Inform. Processing Systems* (Curran Associates, Red Hook, NY), 4708–4717.

Chen W, Lin T, Tan Z, Zhao M, Zhou X (2016) Robust influence maximization. *Proc. 22nd ACM SIGKDD Conf. Knowledge Discovery Data Mining* (ACM, New York), 795–804.

Das A, Kempe D (2008) Algorithms for subset selection in linear regression. *Proc. 40th Annual ACM Symp. Theory Comput.* (ACM, New York), 45–54.

Ene A, Nguyen HL (2016) Constrained submodular maximization: Beyond 1/e. *Proc. 57th Annual Symp. Foundations Comput. Sci.* (FOCS, New Brunswick, NJ), 248–257.

Everingham M, Van Gool L, Williams CKI, Winn J, Zisserman A (2012) The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results [dataset]. Accessed on October 30, 2020, http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html.

Feldman M, Naor J, Schwartz R (2011) A unified continuous greedy algorithm for submodular maximization. *Proc. 52nd Annual Symp. Foundations Comput. Sci.* (FOCS, Palm Springs, CA), 570–579.

Fisher ML, Nemhauser GL, Wolsey LA (1978) An analysis of approximations for maximizing submodular set functions—ii. Balinski ML, Hoffman AJ, eds. *Polyhedral Combinatorics* (Springer, Berlin, Heidelberg), 73–87.

Golovin D, Krause A, Streeter M (2014) Online submodular maximization under a matroid constraint with application to learning assignments. Technical report. California Institute of Technology, Pasadena, CA.

Gomes R, Krause A (2010) Budgeted nonparametric learning from data streams. *Proc. 27th Internat. Conf. Machine Learn.* (Omnipress, Madison, WI), 391–398.

He X, Kempe D (2016) Robust influence maximization. *Proc. 22nd ACM SIGKDD Conf. Knowledge Discovery Data Mining* (ACM, New York), 885–894.

Kalai A, Vempala S (2005) Efficient algorithms for online decision problems. *J. Comput. System Sci.* 71(3):291–307.

Kaufman L, Rousseeuw PJ (1990) *Finding Groups in Data: An Introduction to Cluster Analysis*, vol. 344 (Wiley-Interscience, Hoboken, NJ).

Kazemi E, Zadimoghaddam M, Karbasi A (2018) Scalable deletion-robust submodular maximization: Data summarization with privacy and fairness constraints. Dy J, Krause A, eds. *Proc. 35th Internat. Conf. Machine Learn.* (ICML), vol. 80 (PMLR, Stockholm, Sweden), 2544–2553.

Kempe D, Kleinberg J, Tardos E (2003) Maximizing the spread of influence through a social network. *Theory Comput.* 11(4): 105–147.

Krause A, Guestrin C (2005) Near-optimal nonmyopic value of information in graphical models. *Proc. 21st Conf. Uncertainty Artificial Intelligence* (AUAI Press, Arlington, VA), 324–331.

Krause A, Singh A, Guestrin C (2008a) Near-optimal sensor placements in Gaussian processes: Theory, efficient algorithms and empirical studies. *J. Machine Learn. Res.* 9:235–284.

Krause A, McMahan HB, Guestrin C, Gupta A (2008b) Robust submodular observation selection. *J. Machine Learn. Res.* 9:2761–2801.

Krause A, Rajagopal R, Gupta A, Guestrin C (2009) Simultaneous placement and scheduling of sensors. *Proc. 8th ACM/IEEE Conf. Inform. Processing Sensor Networks* (IPSN) (IEEE, Piscataway, NJ), 181–192.

Lee J, Mirrokni VS, Nagarajan V, Sviridenko M (2009) Non-monotone submodular maximization under matroid and knapsack constraints. *Proc. 41st Annual ACM Symp. Theory Comput.* (STOC) (ACM, New York), 323–332.

Lin H, Bilmes JA (2009) How to select a good training-data subset for transcription: Submodular active selection for sequences. *Proc. 10th Annual Conf. Internat. Speech Comm. Assoc.* (INTERSPEECH, Brighton, UK), 2859–2862.

Minoux M (1978) Accelerated greedy algorithms for maximizing submodular set functions. Stoer J, ed. *Optim. Techniques*. Lecture Notes in Control and Information Sciences, vol. 7 (Springer, Berlin), 234–243.

Mirzasoleiman B, Badanidiyuru A, Karbasi A, Vondrák J, Krause A (2015) Lazier than lazy greedy. *Proc. 29th AAAI Conf. Artificial Intelligence* (AAAI, Menlo Park, CA), 1812–1818.

Mitrovic M, Kazemi E, Zadimoghaddam M, Karbasi A (2018) Data summarization at scale: A two-stage submodular approach. Dy J, Krause A, eds. *Proc. 35th Internat. Conf. Machine Learn.* (PMLR, Stockholm, Sweden), 3593–3602.

Nemhauser GL, Wolsey LA, Fisher ML (1978) An analysis of approximations for maximizing submodular set functions—i. *Math. Programming* 14(1):265–294.

Orlin JB, Schulz AS, Udwani R (2016) Robust monotone submodular function maximization. Louveaux Q, Skutella M, eds. *Proc. 18th Internat. Conf. Integer Programming Combin. Optim.* (Springer, Cham, Switzerland), 312–324.

Powers T, Bilmes J, Krout DW, Atlas L (2016a) Constrained robust submodular sensor selection with applications to multistatic sonar arrays. *Proc. 19th Internat. Conf. Inform. Fusion* (FUSION), 2179–2185.

Powers T, Bilmes J, Wisdom S, Krout DW, Atlas L (2016b) Constrained robust submodular optimization. *30th Conf. Neural Inform. Processing Systems (NIPS 2016), Barcelona, Spain* (Curran Associates, Red Hook, NY).

Rakhlin A (2009) Lecture notes on online learning. Draft, April. Accessed October 30, 2020, http://www-stat.wharton.upenn.edu/~rakhlin/courses/stat991/papers/lecture_notes.pdf.

Schrijver A (2003) *Combinatorial Optimization: Polyhedra and Efficiency*, vol. 24 (Springer Verlag, Berlin, Heidelberg).

Soma T (2019) No-regret algorithms for online k-submodular maximization. Chaudhuri K, Sugiyama M, eds. *Proc. Machine Learn. Res.*, vol. 89, 1205–1214.

Staib M, Wilder B, Jegelka S (2018) Distributionally robust submodular maximization. Chaudhuri K, Sugiyama M, eds. *Proc. Machine Learn. Res.*, vol. 89 (Omnipress, Madison, WI), 506–516.

Streeter M, Golovin D (2008) An online algorithm for maximizing submodular functions. Koller D, Schuurmans D, Bengio Y, Bottou L, eds. *Proc. 21st Internat. Conf. Neural Inform. Processing Systems* (NeurIPS) (Curran Associations, Red Hook, NY), 1577–1584.

Sviridenko M (2004) A note on maximizing a submodular set function subject to a knapsack constraint. *Oper. Res. Lett.* 32(1):41–43.

Tsanas A, Little MA, McSharry PE, Ramig LO (2010) Enhanced classical dysphonia measures and sparse regression for telemonitoring of Parkinson's disease progression. *Proc. 35th IEEE*

Internat. Conf. Acoustics Speech Signal Processing (ICASSP) (IEEE, Piscataway, NJ), 594–597.

Vondrák J (2008) Optimal approximation for the submodular welfare problem in the value oracle model. *Proc. 40th Annual ACM Symp. Theory Comput.* (STOC) (ACM, New York), 67–74.

Wilder B (2017) Equilibrium computation and robust optimization in zero sum games with submodular structure. *Proc. 32nd Conf. Artificial Intelligence* (AAAI Press, Menlo Park, CA), 1274–1281.

Wolsey LA (1982) An analysis of the greedy algorithm for the submodular set covering problem. *Combinatorica* 2(4):385–393.

Zhang M, Chen L, Hassani H, Karbasi A (2019) Online continuous submodular maximization: From full-information to bandit feedback. Wallach H, Larochelle H, Beygelzimer A, d-Alche-Buc F, Fox E, Garnett R, eds. *Proc. 32nd Internat. Conf. Neural Inform. Processing Systems* (NeurIPS) (Curran Associates, Red Hook, NY), 9210–9221.